Sunhayato

フリースケール・セミコンダクタ

HC08 マイコンシリーズ

スタートアップガイド

シリアル・モニタ編

2007年4月15日発行日
REV. 1.10
SG043220

サンハヤト株式会社
本社 〒170-0005 東京都豊島区南大塚3-40-1
☎ 03-3984-7791　FAX. 03-3971-0535
http://www.sunhayato.co.jp

安全上のご注意

このたびは、弊社製品をご使用いただき、誠にありがとうございます。本項では、誤った取り扱いによる事故を未然に防ぐための安全上の注意事項を説明しています。弊社製品をご使用になる前に必ずお読みください。

⚠警告	この表記を無視して誤った取り扱いをすると、死亡や重傷など、人体への重大な障害をもたらす恐れのある内容について示しています。
⚠注意	この表記を無視して誤った取り扱いをすると、軽傷または中程度の障害をもたらす恐れのある内容について示しています。また、本品や本品に接続している機器に損傷を与える可能性がある事項についても示しています。

⚠警告

 プラグをコンセントから抜く	**▶発煙、異臭への対処** 煙がでている、へんな臭いがするなどの異常がありましたら使用を直ちに中止してください。そのまま使用すると、火災、**感電、故障の原因となります。** すぐに電源ケーブルのプラグをコンセントから抜き、煙などの異常が出なくなるのを確認し、販売店などに修理をご依頼ください。
 水場禁止	**▶水分の多いところ、水がかかる場所では、本製品は使用しないで下さい** 風呂場や台所など水分の多いところ、水がかかる場所では、本製品は使用しないで下さい。 火災、感電、故障の原因となります。
 禁止	**▶電源ケーブルの取り扱いに注意してください** 電源ケーブルを傷つけ、破損、加工、無理な曲げ、引っ張り、ねじり、束ねたりしないでください。また、重い物を乗せたり、加熱したりすると**電源ケーブルが破損し、**火災、感電、**故障の原因となります。**
 禁止	**▶医療、軍事、航空宇宙、列車、運送、原子力などの制御設備へは使用しないでください** **医療機器、軍事機器、航空宇宙機器、運送、原子力などの制御設備などの人命に関わ**るシステムへの使用は意図しておりません。
 接触禁止	**▶雷が鳴りはじめたら製品に触れないでください** 近くに雷が発生したときは、製品本体に触れないでください。また、電源プラグをコンセントから抜いてご使用をお控えください。雷によっては、火災、感電、故障の原因となることがあります。

⚠注意

 電源プラグの差し込み	**▶電源プラグは確実にコンセントに差し込んでください** 差し込みが不完全ですと火災、感電、過熱の原因になります。
 分解禁止	**▶分解・改造しないでください** 分解、改造しないでください。怪我、感電、故障の原因となります。本製品の分解、改造による怪我や事故について、当社は責任を負いかねます。
 接触禁止	**▶濡れた手での操作は避けてください** 濡れた手で電源ケーブル・プラグを抜き差ししないでください。また、製品に触れないでください。感電の原因となることがあります。

注意

禁止	**▶以下のような場所では使用しないでください** 本製品を以下のような場所で使用すると、動作不良、故障の原因となります。 ・振動や衝撃が加わる場所 ・直射日光のあたる場所 ・湿気やホコリが多い場所 ・温度差の激しい場所 ・熱を発生するもの（暖房器具など）の近く ・強い磁力、電波が発生するもの（磁石、ディスプレイ、スピーカー、ラジオ、無線機など）の近く ・湿気の多い場所
子供注意	**▶子供の手の届かない場所に置いてください** 本製品に装着されている電子部品など子供が飲み込まないように注意してください。
活線挿抜禁止	**▶通信ケーブルの抜き差しは、電源ＯＦＦにして行ってください** 本品への通信ケーブル類は活線挿抜しないでください。ケーブルの抜き差しは、必ず本製品または相手製品の電源がＯＦＦ状態にて行ってください。故障の原因になることがあります。
安全設計	**▶安全設計をしてください** 本製品を、高度な信頼性を必要とするシステムに使用する場合は、冗長設計、誤動作防止設計など充分な安全設計を必ず行ってください。本製品の故障、傷害により生じるいかなる損害、事故について当社は責任を負いかねます。
保管注意	**▶長期間使用しない場合の保管について** 長期間使用しない場合は、帯電防止袋などに入れ、ホコリなどが入らないようにしてください。ホコリが入ると接触不良などの原因になります。
ホコリ注意	**▶製品の清掃について** 製品にホコリなどが付着すると放熱特性が低下し、故障の原因になりますので、下記の「▶お手入れについて」に従って清掃してください。
薬品注意	**▶お手入れについて** 汚れはやわらかい布によるからぶきをしてください。水、洗剤、ベンジン、シンナーなどの使用は避けてください。基板の洗浄には、サンハヤト製電子機器用洗浄剤をお使いください。
使用注意	**▶故障、破損時の処理について** 本製品が故障もしくは破損した場合は、速やかに使用を中止して販売店などに修理依頼してください。そのまま使用しますと火災、感電、怪我の原因になるおそれがあります。
廃棄注意	**▶本製品の廃棄について** 本製品の廃棄は、各自治体の廃棄ルールに従ってください。詳しくは各自治体にお問い合わせください。
AC100V 以外禁止	**▶日本国内のみで使用してください** 本製品は日本国内の商用 AC100V 電源仕様です。海外では使用できません。AC100V 以外では絶対に使用しないでください。

本資料についてのご注意

本資料について

- 本資料は、電子工作や電子回路、パーソナルコンピュータの操作について一般的な知識をお持ちの方を対象にしています。
- 本資料を元に操作するには、フリースケール・セミコンダクタ社製 HC08 シリーズマイコンについての知識や開発環境などが必要です。
- Microsoft®、Windows® は米国 Microsoft 社の米国およびその他の国における登録商標です。
- その他、記載されている製品名は各社の商標または登録商標です。

本資料のご利用にあたって

- この取扱説明書に掲載している内容は、お客様が用途に応じた適切な製品をご購入頂くことを目的としています。その使用により当社及び第三者の知的財産権その他の権利に対する保証、又は実施権の許諾を意味するものではありません。また、権利の侵害に関して当社は責任を負いません。
- 本資料に記載した情報を流用する場合は、お客様のシステム全体で充分評価し適用可能かご判断願います。当社では適用可能判断についての責任は負いません。
- 本資料に記載してある内容は、一般的な電子機器（学習教材、事務機器、計測機器、パーソナル機器、コンピュータ機器など）に使用されることを目的としています。高い品質や信頼性が要求され、故障や誤作動が直接人命を脅かしたり人体に危害を及ぼす恐れのある、医療、軍事、航空宇宙、原子力制御、運輸、移動体、各種安全装置などの機器への使用は意図も保証もしておりません。
- この取扱説明書の一部、又は全部を当社の承諾なしで、いかなる形でも転載又は複製されることは堅くお断りします。
- 全ての情報は本資料発行時点のものであり、当社は予告なしに本資料に記載した内容を変更することがあります。
- この資料の内容は慎重に制作しておりますが、万一記述誤りによってお客様に損害が生じても当社はその責任を負いません。
- 本資料に関してのお問合せ、その他お気付きの点がございましたら、当社までお問合せください。
- 本資料に関する最新の情報はサンハヤト株式会社ホームページ（http://www.sunhayato.co.jp/）に掲載しております。

このマニュアルについて

本スタートアップガイドでは、サンハヤトHC08マイコンシリーズのプログラム開発を、フリースケール・セミコンダクタ社が無償で配布しているCodeWarrior Development Studio for Freescale HC(S)08　Version 5.1 Special Edition（以下CodeWarrior）で行い、シリアル・モニタでデバッグを行う場合の手順、設定方法などを説明したものです。CodeWarriorの詳細な使用方法や注意事項につきましては、フリースケール・セミコンダクタ社発行のユーザーズ・マニュアルを参照してください。

■該当するサンハヤトHC08マイコンシリーズ製品

- ターゲットマイコン　：SF9S08Cマイコン IC（フリースケール・セミコンダクタ社 MC9S08QG8）
- 学習／評価カード　　：CT-298

■開発に必要なもの

- 以下の条件を満たすホストパーソナルコンピュータ（以下ホストPC）

PC本体	400MHz、Pentium® Ⅱ　あるいはAMD-K6®クラス以上を搭載したIBM PC/AT互換機
OS	Windows 98 SE, Windows 2000, Windows XP（Windows XP推奨）
メモリ	128MB以上
HDD	500MB以上のハードディスクの空き
入力デバイス	マウス、またはマウス相当のポインティングデバイス
インターフェイス	USB1.1 フルスピード、あるいは2.0

- CT-298
- USBケーブル

■このマニュアルで使用しているツール類

このスタートアップガイドでは、以下のOS、ツールのバージョンで動作したものとして説明しています。

ホストPCのOS	Windows XP Professional
統合開発環境（コンパイラ含む）	CodeWarrior Development Studio for Freescale HC(S)08 version5.1 Special Edition

CodeWarriorのバージョンは「Help」＞「About Freescale CodeWarrior IDE」＞「Installed Products」で確認することができます。

■このマニュアルで紹介している各社サイト、ツールについて

本スタートアップガイドで紹介している各社サイトやツールは、本スタートアップガイド発行時のものを掲載しております。各社サイトの構成やツールのファイル名などは、変更されている場合がありますのでご了承ください。

目次

このマニュアルについて …… 5

■該当するサンハヤト HC08 マイコンシリーズ製品 …… 5
■開発に必要なもの …… 5
■このマニュアルで使用しているツール類 …… 5
■このマニュアルで紹介している各社サイト、ツールについて …… 5

1. 開発の流れ …… 8

1.1 開発環境 …… 8
1.2 シリアル・モニタによる開発環境 …… 9
■特長 …… 9
■起動時の動作 …… 9
1.3 シリアル・モニタによる制限事項 …… 10
■メモリの制限 …… 10
■マイコン機能の制限 …… 11
■ベクタ・アドレスの制限 …… 11
■レジスタ操作の制限 …… 12
■動作クロックの制限 …… 12
■ STOP 命令の制限 …… 12

2. 開発環境を整えよう …… 13

2.1 開発に必要なソフトウェアをダウンロードしよう …… 13
■ CodeWarrior …… 13
コラム：仮想 COM ポートドライバについて …… 14
■仮想 COM ポートドライバ …… 14
2.2 インストールの手順 …… 15

3. プログラムをつくろう …… 19

3.1 LED 点滅プログラムをつくる …… 19
3.2 プロジェクトの準備 …… 20
■プロジェクトウィザード …… 20
■日本語表示の設定 …… 25
3.3 Device Initialization で初期設定関数を生成する …… 26
■パッケージを設定する …… 26
■ CPU の設定をする …… 27
コラム：内部クロックについて …… 28
■ PTB の設定をする …… 30
■ TPM の設定をする …… 33
コラム：TPM（Timer/Pulse-With Modulator）のカウント値について …… 34
■初期設定関数を生成する …… 36

3.4 シリアル・モニタに合わせて各ファイルの修正をする …… 38
■初期設定関数ファイル「MCUinit.c」を修正する …… 38
3.5 プログラムの編集をする …… 40
■必要のないコード、コメントを削除する …… 40
■ main 関数をつくる …… 40
■ 1 秒計測関数、LED 出力関数をつくる …… 41
コラム：TPM の割り込みについて …… 41
3.6 プログラムをコンパイルする …… 42

4. プログラムをうごかそう …… 43
4.1 CodeWarrior からデバッガを起動する …… 43
4.2 プログラムの実行と停止 …… 46
4.3 プログラムをステップ実行する …… 46
4.4 プログラムの開始アドレスを変更する …… 47
4.5 プログラムの停止条件を設定する …… 48
■ブレークポイントを設定する …… 48
■行にトリガを設定する …… 49
■メモリにトリガを設定する …… 50
4.6 メモリを参照する …… 51
4.7 プロジェクトを閉じる …… 52
4.8 プロジェクトを開く …… 52

5. トラブルシューティング …… 54
5.1 デバッガと通信できない …… 54
■シリアル・モニタがデバッグモードで動作していない …… 54
■ COM ポート番号が合っていない …… 54
■仮想 COM ポート番号が設定可能範囲以外の場合 …… 54
5.2 仮想 COM ポート番号の確認 …… 55
5.3 仮想 COM ポート番号の割り当て …… 56

改訂履歴 …… 58

1. 開発の流れ

1.1 開発環境

サンハヤト HC08 シリーズマイコン製品でプログラムを開発する場合、以下の開発ツール（フリースケール・セミコンダクタ社）、仮想 COM ポートドライバ（FTDI 社）を使用します。ここで紹介している各ソフトウェアは無償で提供されています。

■ 統合開発環境（アセンブラ、コンパイラ、シリアルデバッガ）：CodeWarrior
CodeWarrior では、プログラム作成からコンパイル（アセンブル）、デバッグまでの各フェーズを同一 GUI で行うことができます。

■ FTDI 社 仮想 COM ポートドライバ
USB －シリアル変換 IC（FT232R）をホスト PC の仮想 COM ポートで使用するためのドライバソフトウェアです。

開発フロー（イメージ図）を以下に示します。

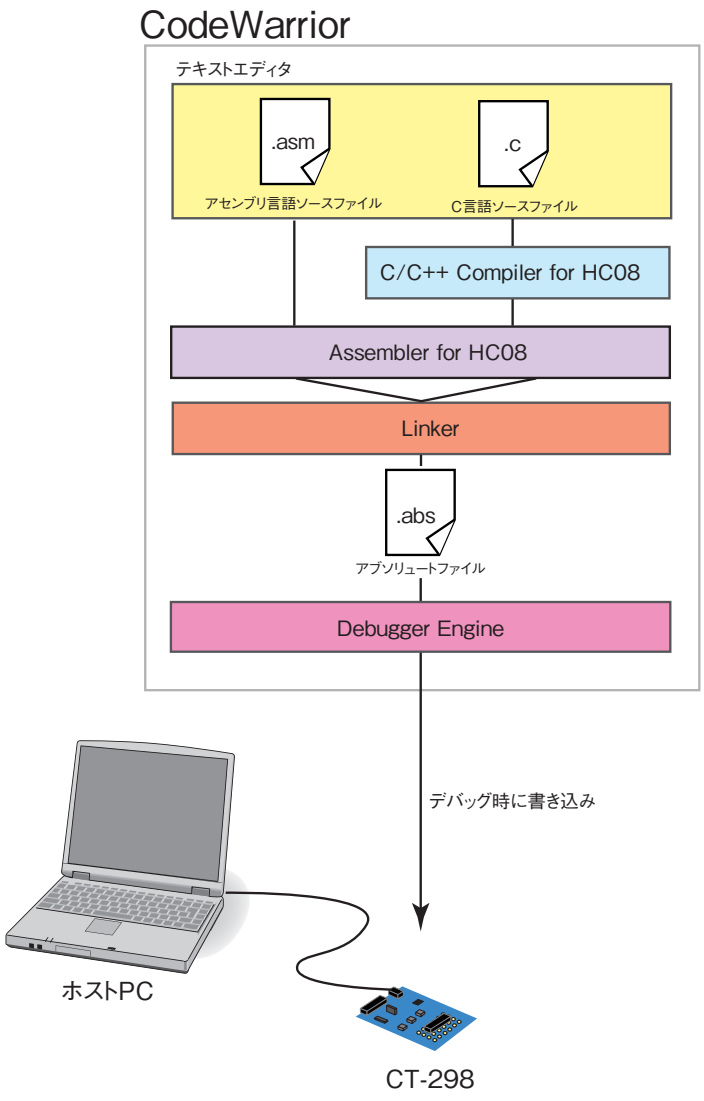


図1-1 CodeWarrior 開発フロー（イメージ図）

1.2 シリアル・モニタによる開発環境

■特長

サンハヤト ミニ評価カード CT-298 に搭載している SF9S08C マイコンのフラッシュメモリには、あらかじめデバッグ用のシリアル・モニタ（プログラム）が書き込まれており、エミュレータ等を使用しなくても USB ケーブルでホスト PC と接続するだけでユーザープログラムを書き込み、実機デバッグをすることができます。

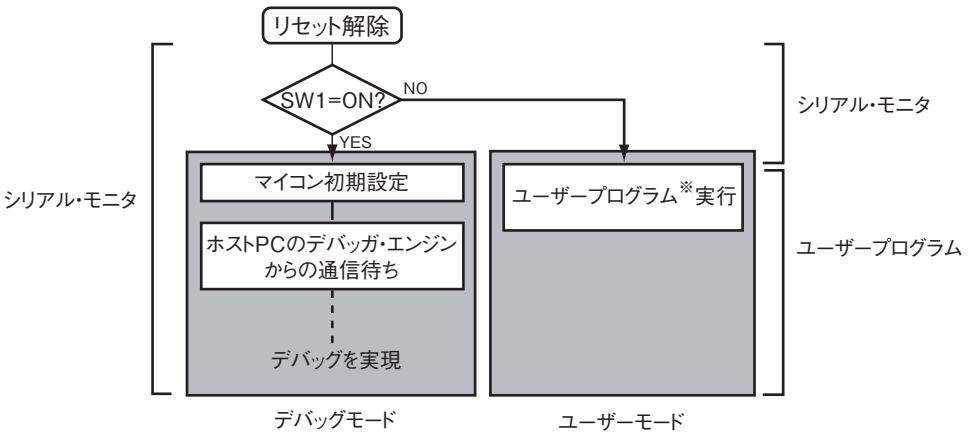
シリアル・モニタが書き込まれているフラッシュ・メモリの領域はプロテクトされており、誤って消去されることはありません。（BDM ツールでは書き換えることができます。）

SF9S08C マイコンに書き込まれたシリアル・モニタは、ホスト PC のデバッガ・エンジンと通信して、デバッグを実現します。SF9S08C マイコンはデバッグ中、シリアル・モニタとユーザープログラムを交互に実行することになります。この原理をふまえ、注意事項に留意することで、シリアル・モニタを使用した開発環境を最大限に活用することができます。

■起動時の動作

SF9S08C マイコンは電源投入時あるいはリセットボタンが押された後、シリアル・モニタが動作します。シリアル・モニタは、CT-298 上の SW1（SF9S08C マイコンの PTA2 ポートへ接続）が押されていればデバッグモードへ移行します。SW1 が押されていなければ、ユーザープログラム（ユーザーモード）を実行します。

また、SF9S08C マイコンのユーザープログラム領域（シリアル・モニタ領域とは別領域です）にはあらかじめ 3 軸加速度センサーモジュール（MM-2860）デモプログラムが書き込まれており、CT-298 に MM-2860 を搭載している場合はすぐに評価することができます。デモプログラムの評価方法については別紙の「アプリケーションノート　3 軸加速度センサーモジュール デモプログラムの評価方法」を参照してください。



※製品初期状態は、3軸加速度センサーモジュールデモプログラムが書き込まれています。

図1-2 SF9S08Cマイコン起動時の動作

1.3 シリアル・モニタによる制限事項

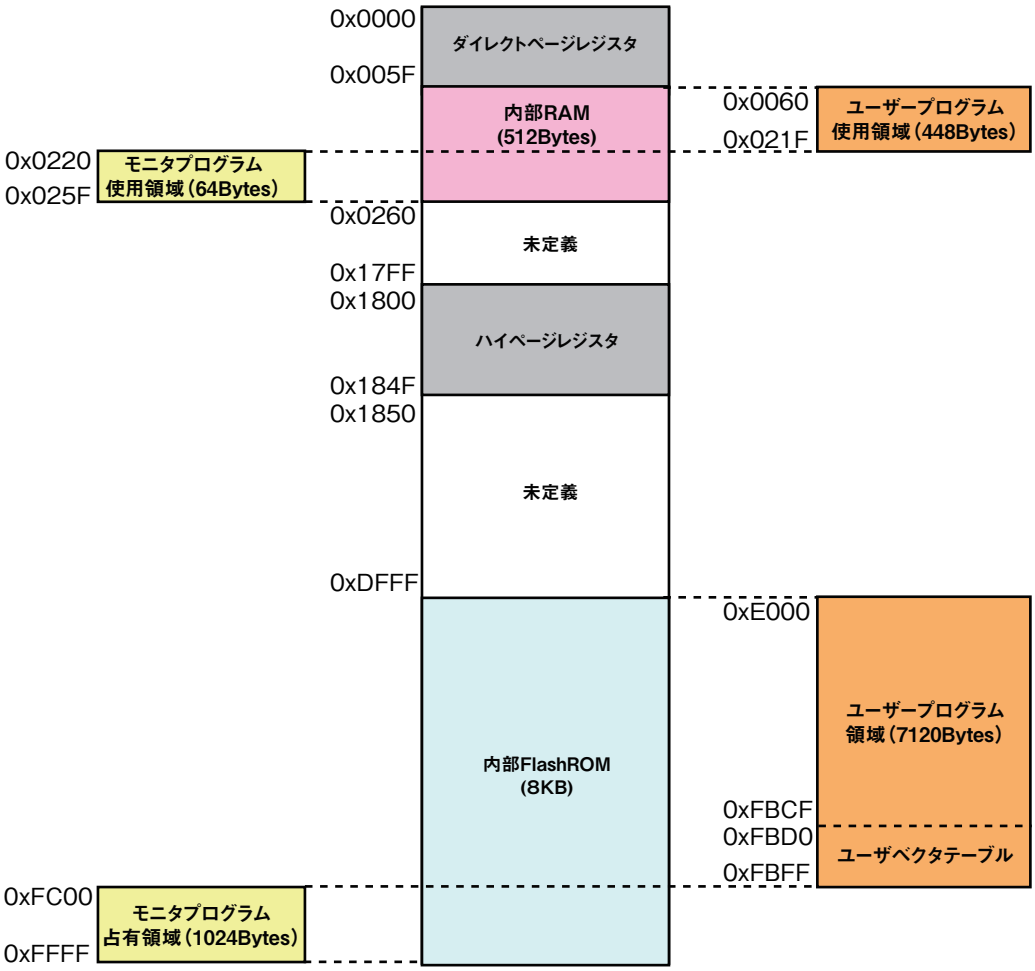
シリアル・モニタによるデバッグを行う場合、以下の制限事項が発生します。

■メモリの制限

以下の領域は、シリアル･モニタが使用、あるいは書き込まれている領域ですので、ユーザーは使用できません。

- 0x0220 ～ 0x025F：シリアル・モニタが使用する領域
- 0xFC00 ～ 0xFFFF：シリアル・モニタが書き込まれている領域

以下にシリアル・モニタでデバッグ時のメモリ・マップを示します。



ダイレクトページレジスタ、ハイページレジスタには主に周辺機能のレジスタが配置されています。

内部 Flash ROM は、書き込み / 消去がおよそ 100,000 回可能です。（標準温度 25℃、電源電圧 1.8 ～ 3.6V）

図1-3 デバッグ時のメモリマップ

■マイコン機能の制限

SCI（Serial Communication Interface：シリアル通信）、SWI（Software Interrupt：ソフトウェア割り込み）はシリアル・モニタが使用しますので、ユーザーは使用できません。また、シリアル・モニタの設定によりデバッグ中は COP（ウォッチドッグ・タイマ機能）は使用できません。

■ベクタ・アドレスの制限

通常のベクタ・アドレス領域（0xFFD0 ～ 0xFFFF）はシリアル・モニタが使用します。ユーザーが使用するベクタ・アドレスは 0xFBD0 から設定してください。また、SCI（シリアル受信 / 送信割り込み）、SWI（ソフトウェア割り込み）はシリアル・モニタが使用しますので $FFFF を設定してください。（ユーザーは使用できません。）リセット・ベクタには $FFxx 以外のアドレスを設定してください。

表1-1 デバッグ時のベクタ・アドレス

通常のベクタ・アドレス	割り込みソース	ユーザーベクタ・アドレス	設定内容
0xFFD0,0xFFD1	RTI（リアルタイム割り込み）	0xFBD0,0xFBD1	
0xFFD2 ～ 0xFFD5	Reserved	0xFBD2 ～ 0xFBD5	Reserved
0xFFD6,0xFFD7	ACMP（アナログ・コンパレータ）	0xFBD6,0xFBD7	
0xFFD8,0xFFD9	AD コンバータ	0xFBD8,0xFBD9	
0xFFDA,0xFFDB	KBI（キーボード割り込み）	0xFBDA,0xFBDB	
0xFFDC,0xFFDD	IIC	0xFBDC,0xFBDD	
0xFFDE,0xFFDF	SCI 送信	0xFBDE,0xFBDF	$FFFF にしてください
0xFFE0,0xFFE1	SCI 受信	0xFBE0,0xFBE1	$FFFF にしてください
0xFFE2,0xFFE3	SCI エラー	0xFBE2,0xFBE3	$FFFF にしてください
0xFFE4,0xFFE5	SPI	0xFBE4,0xFBE5	
0xFFE6,0xFFE7	MTIM オーバーフロー	0xFBE6,0xFBE7	
0xFFE8 ～ 0xFFEF	Reserved	0xFBE8 ～ 0xFBEF	Reserved
0xFFF0,0xFFF1	TPM オーバーフロー	0xFBF0,0xFBF1	
0xFFF2,0xFFF3	TPM チャネル 1	0xFBF2,0xFBF3	
0xFFF4,0xFFF5	TPM チャネル 0	0xFBF4,0xFBF5	
0xFFF6,0xFFF7	Reserved	0xFBF6,0xFBF7	Reserved
0xFFF8,0xFFF9	LVD（低電圧検出）	0xFBF8,0xFBF9	
0xFFFA,0xFFFB	IRQ（外部割込み）	0xFBFA,0xFBFB	
0xFFFC,0xFFFD	SWI（ソフトウェア割り込み）	0xFBFC,0xFBFD	$FFFF にしてください
0xFFFE,0xFFFF	Reset	0xFBFE,0xFBFF	$FFxx 以外にしてください

■レジスタ操作の制限

以下の機能に関連するレジスタの内容をユーザープログラムで変更しないでください。

- SCI（Serial Communication）に関連するレジスタ
- BDC（Background Debug Controller）に関連するレジスタ

デバッグモードでは、以下に示すレジスタは初期化されます。ユーザープログラムで変更しないでください。

デバッグモードとユーザーモードで同じ条件（システム・クロック、バス周波数、Reset 端子の設定）で動作させるためには、ユーザープログラムでも同じ設定を行ってください。

表1-2 シリアル・モニタが設定するレジスタと設定値

レジスタ	リセット後の値	シリアル・モニタが設定する値
SOPT1 (COP and STOP enable controls)	11010011（PowerOnReset、Low-voltage Reset） 1101001u（Reset） u=Unaffected	01110011 ¦¦¦ ¦¦ ¦¦¦ ¦+-RSTPE --- Reset pin enabled ¦¦¦ +--BKGDPE -- BKGD pin enabled ¦¦+------STOPE --- STOP allowed ¦+-------COPT ---- long timeout 2^18 +--------COPE ---- COP off
ICSC1 (ICS Control1)	00000100 ¦¦¦¦¦¦¦¦ ¦¦¦¦¦¦¦+-IREFSTEN - Int ref clk dis in stop ¦¦¦¦¦¦+--IRCLKEN - ICSIRCLK inactive ¦¦¦¦¦+---IREFS --- Int ref clk selected ¦¦+++----RDIV --- Divides ref clk by 1 ++------CLKS --- Output of FLL is selected	リセット後の値と同じ
ICSC2 (ICS Control2)	01000000	00000000 ¦¦¦¦¦¦¦¦ ¦¦¦¦¦¦¦+-EREFSTEN - Ext ref disable in stop ¦¦¦¦¦¦+--ERCLKEN - ICSERCLK inactive ¦¦¦¦¦+---EREFS --- Ext OSC (don't care) ¦¦¦¦+----LP --- FLL enable ¦¦¦+-----HGO --- Low power ¦¦+------RANGE --- Low range ++-------BDIVE --- BDIV=1
ICSTRM (ICS Trim Register)	10000000	FFAF 番地の値
ICSSC (ICS Status & Contorol)	00000000	FFAE 番地の値

■動作クロックの制限

デバッグモードでは、コア・クロックは内部クロックを利用した FEI モード（トリミング済み）、16MHz となり、バス・クロックは 8MHz となります。

ユーザープログラムで上記以外の設定を行わないでください。また、ユーザープログラムでクロックの設定を行わない場合、デバッグモード時はシリアル・モニタが設定したクロックで動作し、ユーザーモード時は SF9S08C マイコンのデフォルトのクロック設定（コア・クロック：FEI モード 16MHz トリミング済み、バス・クロック：4MHz）で動作しますのでご注意ください。

■STOP命令の制限

STOP 命令は STOP1 モードの場合使用できません。STOP2 モードと STOP3 モードに設定した場合は STOP 命令が利用できます。但し、デバッガで STOP 命令をステップ実行（Trace 実行）すると、STOP モードによりクロックが停止するためデバッガとの通信が途切れてしまい、再接続する必要があります。Go コマンドの場合は問題ありません。

2. 開発環境を整えよう

2.1 開発に必要なソフトウェアをダウンロードしよう

■CodeWarrior

フリースケール・セミコンダクタ社のページ（http://www.freescale.co.jp/）へアクセスしてください。トップページの「製品」メニューより「CodeWarrior 開発ツール」>「CodeWarrior Products」>「HC(S)08（英語）」で開いたページのリンク「CodeWarrior Development Studio for HC08 Microcontrollers (Special Edition) 」をクリックしてください。ユーザ ID、パスワードのページが開きます。（ユーザ ID、パスワードを取得するには、「フリースケール・フリーク」メンバー登録が必要です。）登録済みのユーザ ID、パスワードを入力するとダウンロードが始まります。

※ 上記リンク、また下図の画像は本スタートアップガイド発行時のものです。フリースケール・セミコンダクタ社のサイト構成の変更などで内容が違っている場合があります。該当するページが見つからない場合は、キーワード「CodeWarrior for HC(S)08/RS08」などで検索してください。

図2-1 CodeWarriorのダウンロード

■仮想COMポートドライバ

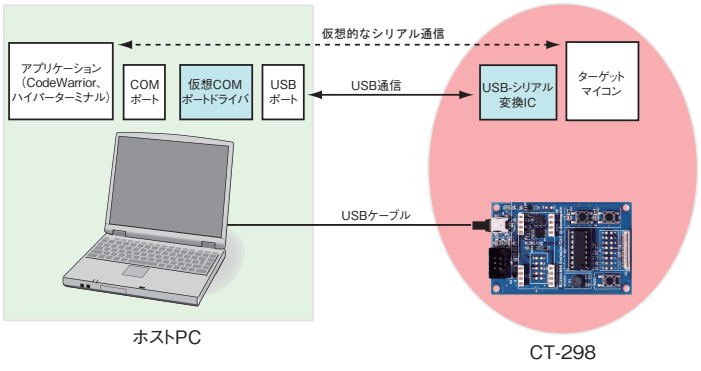
FTDI 社 FTDI 社のサイト（http://www.ftdichip.com/）より、トップページ >Drivers>VCP>FT232R（OS はお使いのホスト PC の環境に合わせて選んでください。）をダウンロードしてください。ファイルは解凍して任意のフォルダに保存してください。

図2-2 仮想COMポートドライバのダウンロード

仮想COMポートドライバについて

CT-298 には FTDI 社の USB- シリアル変換 IC が搭載されており、対応する仮想 COM ポートドライバがインストールされているホスト PC と仮想的なシリアル通信を行うことができます。

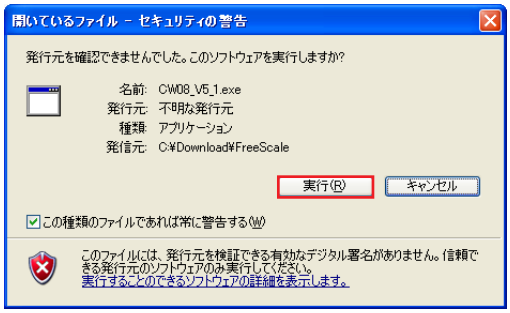
仮想的なシリアル通信とはこの場合、実際は USB ケーブルで接続して USB 通信を行っていますが、マイコン - アプリケーション間ではあたかもシリアル通信を行っているような通信のしくみをいいます。



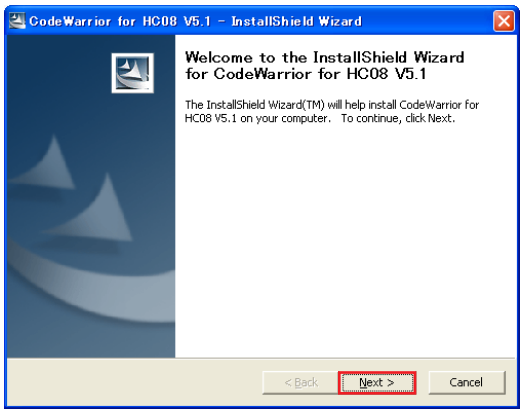
2.2 インストールの手順

1 CodeWarrior のインストール

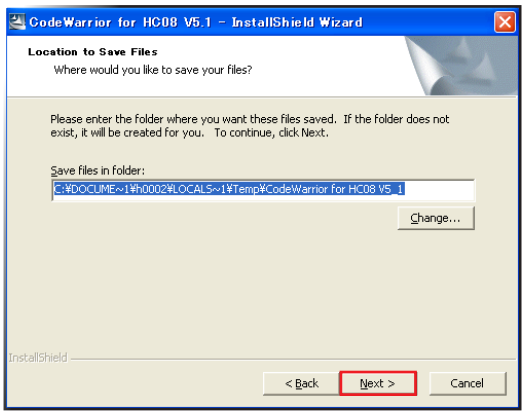
インストーラファイル「CW08_V5_1.exe」をダブルクリックし、「セキュリティの警告」が出た場合は「実行」をクリックしてください。



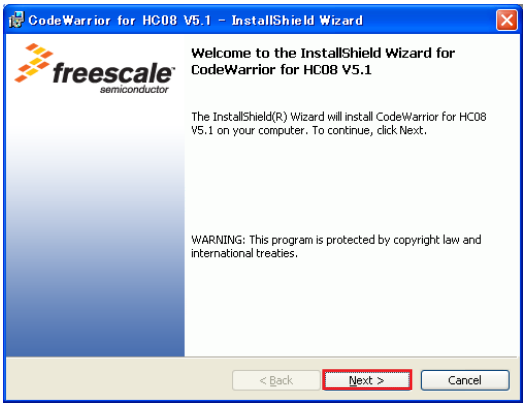
「Next」をクリックしてください。


CodeWarrior for HC08 V5.1 - InstallShield Wizard
Welcome to the InstallShield Wizard for CodeWarrior for HC08 V5.1
The InstallShield Wizard(TM) will help install CodeWarrior for HC08 V5.1 on your computer. To continue, click Next.
< Back
Next >
Cancel

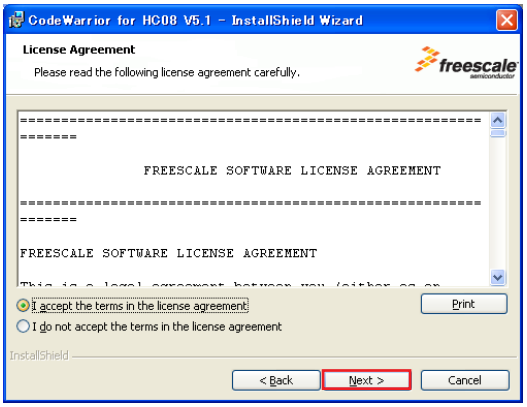
ファイルの解凍先を指定します。解凍後はインストーラが自動的に起動します。


CodeWarrior for HC08 V5.1 - InstallShield Wizard
Location to Save Files
Where would you like to save your files?
Please enter the folder where you want these files saved. If the folder does not exist, it will be created for you. To continue, click Next.
Save files in folder:
C:¥DOCUME~1¥h0002¥LOCALS~1¥Temp¥CodeWarrior for HC08 V5_1
Change...
InstallShield
< Back
Next >
Cancel

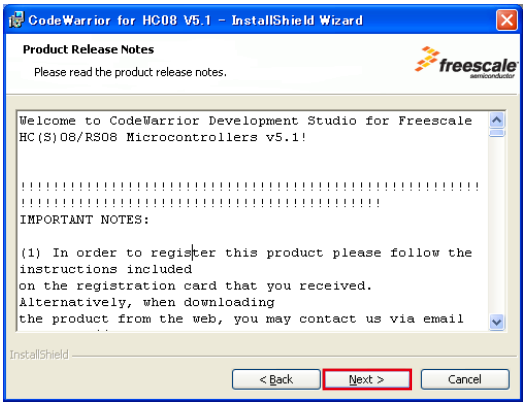
「Next」をクリックしてください。


CodeWarrior for HC08 V5.1 - InstallShield Wizard
freescale semiconductor
Welcome to the InstallShield Wizard for CodeWarrior for HC08 V5.1
The InstallShield(R) Wizard will install CodeWarrior for HC08 V5.1 on your computer. To continue, click Next.
WARNING: This program is protected by copyright law and international treaties.
< Back
Next >
Cancel

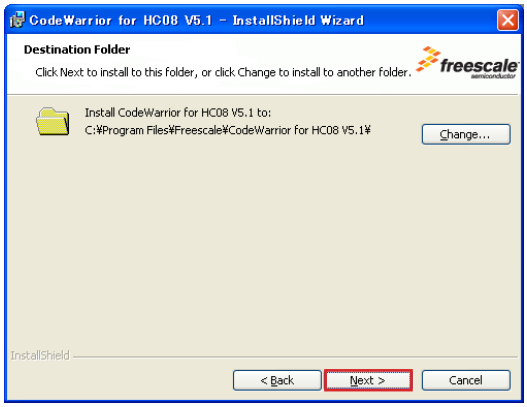
「License Agreement」の内容を確認したら「I accept the terms of the license agreement」にチェックを入れて「Next」をクリックしてください。


CodeWarrior for HC08 V5.1 - InstallShield Wizard
License Agreement
Please read the following license agreement carefully.
freescale semiconductor
FREESCALE SOFTWARE LICENSE AGREEMENT
FREESCALE SOFTWARE LICENSE AGREEMENT
I accept the terms in the license agreement
I do not accept the terms in the license agreement
Print
InstallShield
< Back
Next >
Cancel

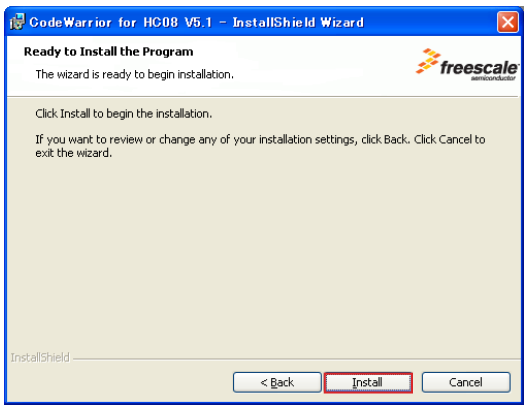
リリースノートが表示されます。「Next」をクリックしてください。


CodeWarrior for HC08 V5.1 - InstallShield Wizard
Product Release Notes
Please read the product release notes.
freescale semiconductor
Welcome to CodeWarrior Development Studio for Freescale HC(S)08/RS08 Microcontrollers v5.1!
IMPORTANT NOTES:
(1) In order to register this product please follow the instructions included
on the registration card that you received.
Alternatively, when downloading
the product from the web, you may contact us via email
InstallShield
< Back
Next >
Cancel

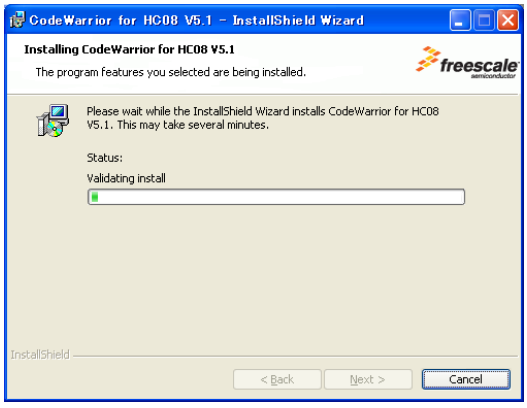
インストール先はデフォルトのままにしておくことをお勧めします。
「Next」をクリックしてください。



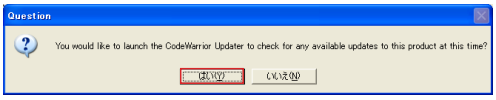
インストールする内容、インストール先を確認して、「Next」をクリックしてください。



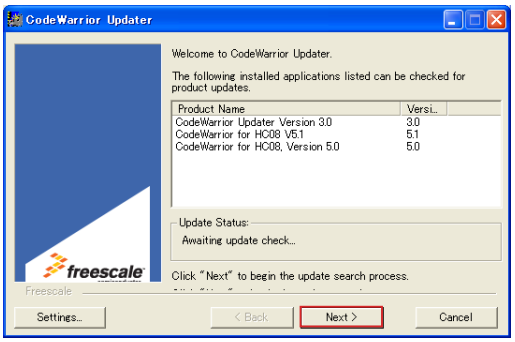
インストールが始まります。



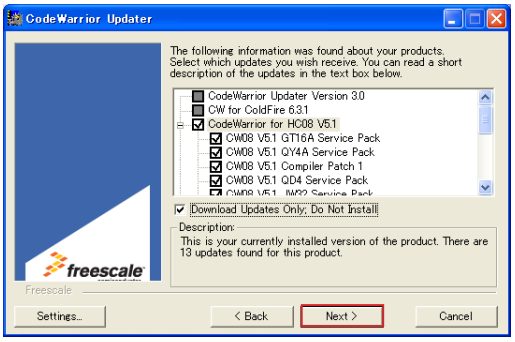
アップデートを行うかの確認メッセージが出ます。「はい」をクリックすると、アップデートの確認が行われます。



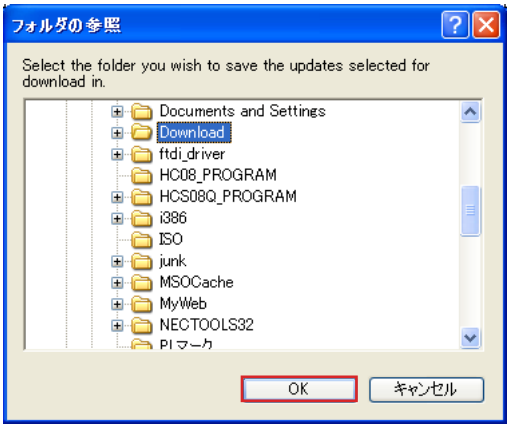
「Next」をクリックしてください。アップデータによってアップデートできるアプリケーションが表示されます。



アップデートがあった場合、チェックボックスがチェック可能となります。チェックすると、関連するアップデートがダウンロードされます。
「Next」をクリックしてください。

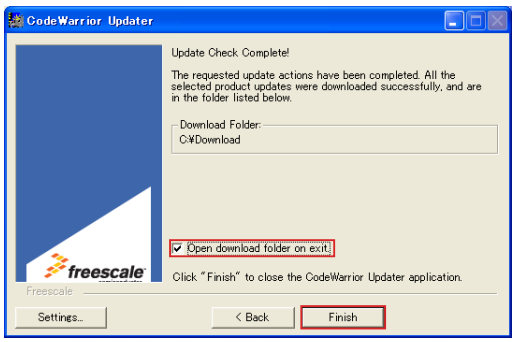


アップデートファイルのダウンロード先を指定します。ここでは C: ¥Download としています。

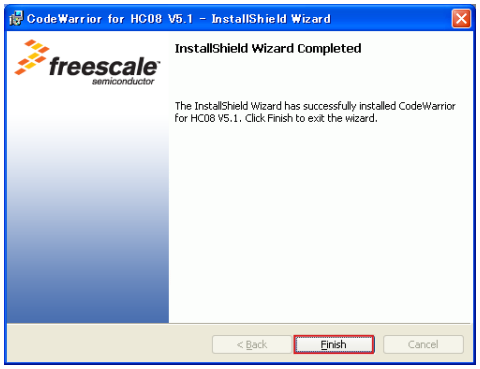


アップデートファイルがダウンロードされます。

「Open downloaded folder on exit」をチェックして「Finish」をクリックすると、ウィンドウが閉じたあとに、ダウンロードしたフォルダが開きます。

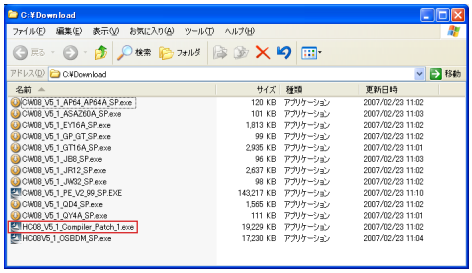


インストールが終了しました。「Finish」をクリックしてください。

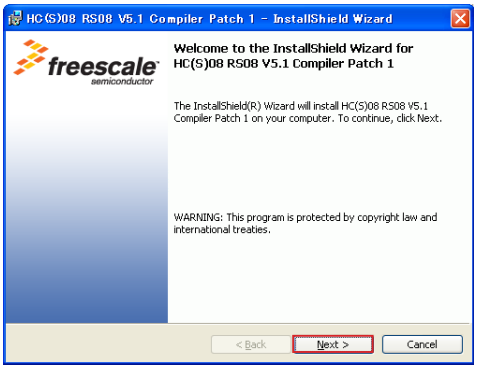


アップデートは、ここでは「HC08_V5_1_Compiler_Patch_1.exe」のみ行うこととします。アップデートには、MC9S08QG8 マイコン以外のマイコンをサポートするためのものも含まれており、全てを実行する必要はありませんが、判断がつかないようでしたら実行しておくことをお勧めします。

「HC08_V5_1_Compiler_Patch_1.exe」ファイルをダブルクリックしてください。

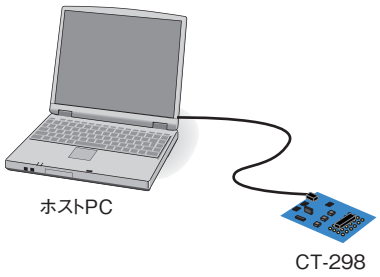


アップデートが実行されます。あとは画面の指示に従って進めてください。

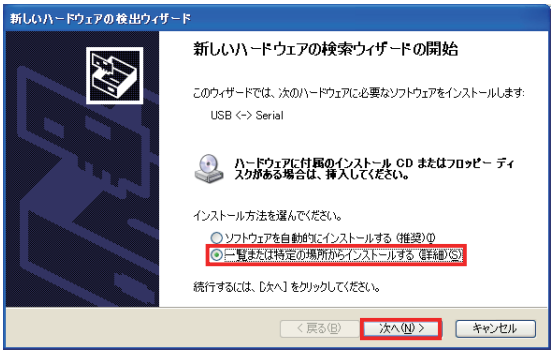


2 仮想 COM ドライバのインストール

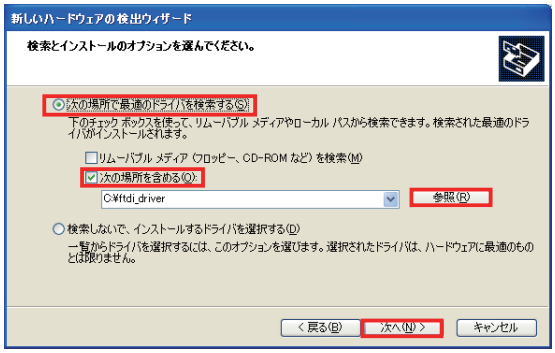
評価カード CT-298 とホスト PC を USB ケーブルで接続してください。

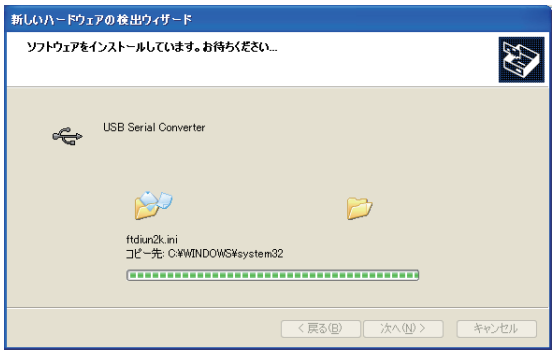


「新しいハードウェアの検索ウィザードの開始（USB ⇔ Serial）」ウィンドウが開きます。「一覧または特定の場所からインストールする（詳細）」にチェックが入った状態で「次へ」をクリックしてください。

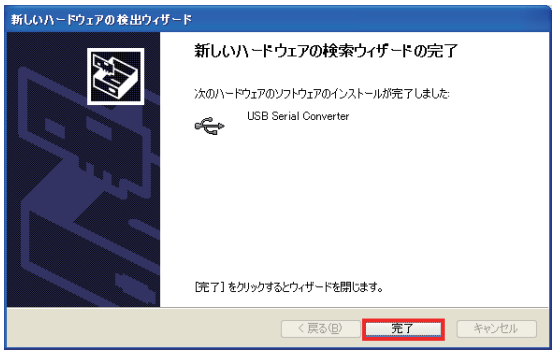


ドライバファイルのフォルダを指定します。ドライバが検索できたらインストールを開始します。

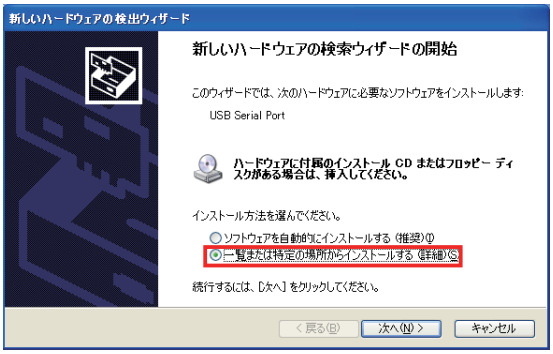




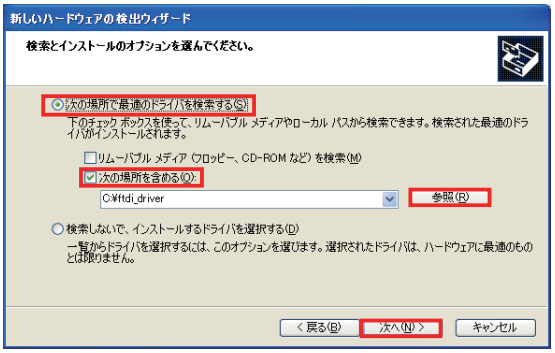
「完了」をクリックしてください。



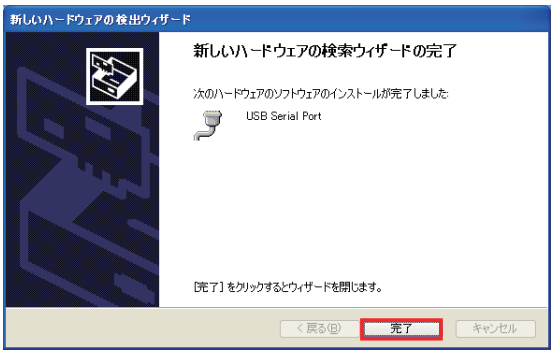
「新しいハードウェアの検索ウィザードの開始(USB Serial Port)」ウィンドウが開きます。「一覧または特定の場所からインストールする（詳細)」にチェックが入った状態で「次へ」をクリックしてください。



ドライバファイルのフォルダを指定します。ドライバが検索できたらインストールを開始します。



「完了」をクリックしてください。



3. プログラムをつくろう

3.1 LED 点滅プログラムをつくる

評価カード CT-298 の 2 つの LED を点滅させる簡単なプログラムをつくってみましょう。

LED1 から約 0.1 秒間隔で交互に点滅するものとします。LED は SF9S08C マイコンの汎用 I/O ポート（PTB：PARALLEL INPUT/OUTPUT CONTROL MODULE）に接続されています。0.1 秒の計測には SF9S08C マイコンの 16 ビットタイマ（TPM：16BIT TIMER/PWM MODULE）を使用します。

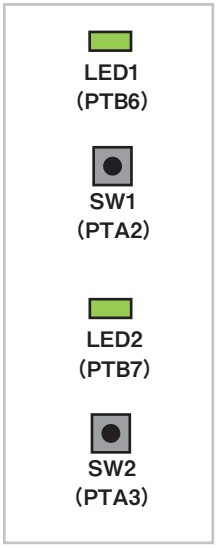


図3-1 CT-298のLED

各 LED の回路構成は以下のようになります。出力ポートより“L”出力で点灯します。

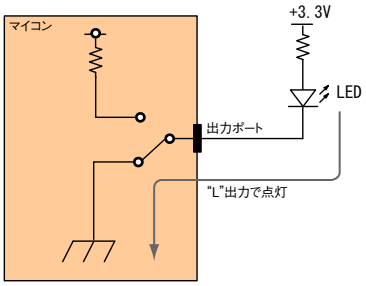


図3-2 LED回路（イメージ図）

3.2 プロジェクトの準備

Windows の「スタート」メニューから「すべてのプログラム」>「Freescale CodeWarrior」>「CW08 V5.1」>「CodeWarrior IDE」をクリックして CodeWarrior を起動してください。

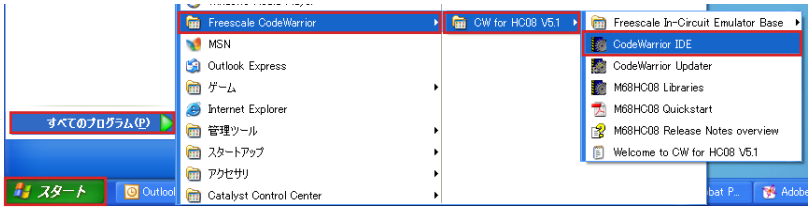


図3-3 CodeWarrior：起動

CodeWarrior では、プログラムのソースファイル、アセンブル、コンパイル、リンクや、使用するデバッガ等の情報をプロジェクトという単位で扱います。新しくプログラムをつくる場合は、プロジェクトを作成します。

また、プロジェクト作成時の設定で、スタートアッププログラム、プログラムの雛形となる main 関数ファイルを作成することができます。

■プロジェクトウィザード

「Startup」ウィンドウが開いたら「Create New Project」をクリックしてください。プロジェクトウィザード（プロジェクト作成のための設定）が始まります。

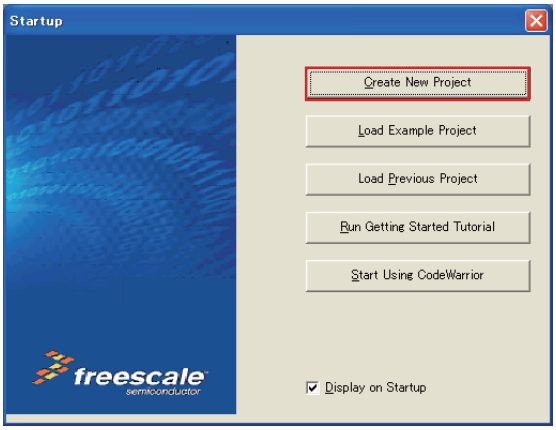


図3-4 CodeWarrior：新規プロジェクト作成

「Device and Connection」フェーズでは、ターゲットマイコンの型名と書き込み、デバッグ時の通信方法を設定します。以下のように設定し、「次へ」をクリックしてください。

- Select the derivative…：HCS08>HCS08QG Family>MC9S08QG8 を選択します。
- Connections：HCS08 Serial Monitor を選択します。

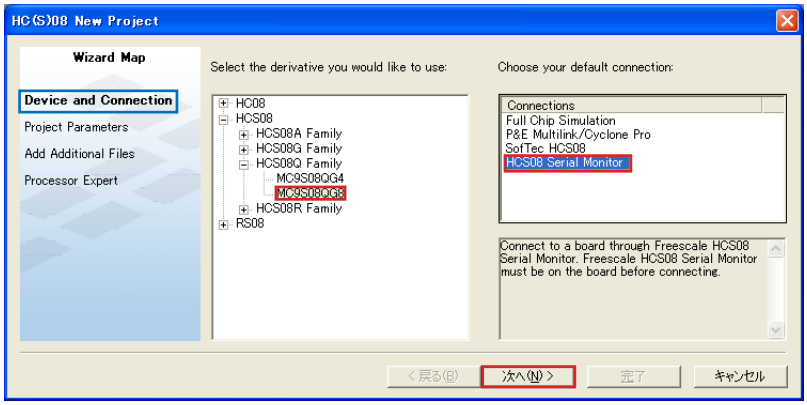


図3-5 CodeWarrior：Device and Connection

「Project Parameters」フェーズでは開発言語、プロジェクト名と保存先を設定します。プログラムはC言語で開発するものとします。以下のように設定し、「次へ」をクリックしてください。

- Please choose the set of languages…：「C」にチェックを入れます。
- Project name：プロジェクトの名称を決めます。ここでは「LED_PICOPICO_F」とします。
- Location：プロジェクトの保存先を指定します。
  「Set」ボタンをクリックして任意のフォルダ内で保存先のフォルダ名を入れて保存するとプロジェクトフォルダが作成されます。ここではC:¥HC08_PROGRAMフォルダ以下に「LED_PICOPICO_F」というプロジェクト名と同じ名前のフォルダを作成し、保存します。

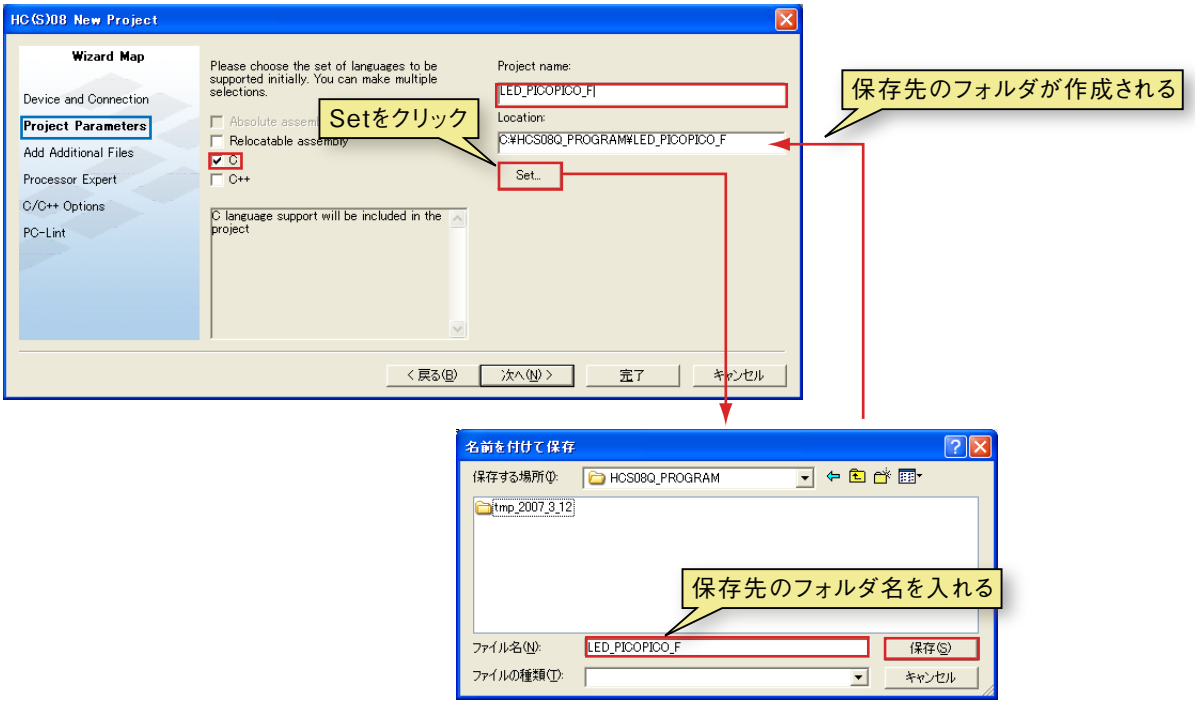


図3-6 CodeWarrior：Project Parameters

「Add Additional Files」フェーズでは、プロジェクトにソースファイルなどをコピーしたり、新しく main 関数を含むファイルを作成するかどうかの設定をします。ここではファイルのコピーはしないこととします。また、メイン関数を含むファイルを作成するので、「Create main.c/main.asm file」にチェックが入っていることを確認して「次へ」をクリックしてください。

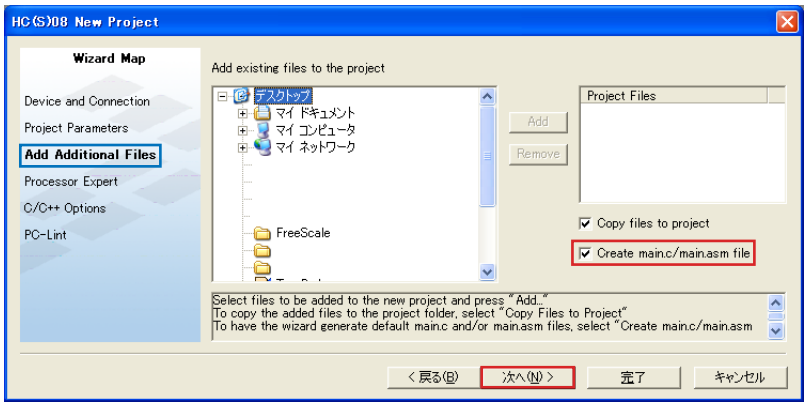


図3-7 CodeWarrior：Add Additional Files

「Processor Expert」フェーズでは、マイコンや周辺機能の初期設定の関数を作成するかどうかを設定します。「None」は何も作成しません。「Device Initialization」はマイコンや周辺機能の初期設定関数を作成します。「Processor Expert」では初期設定関数に加え、周辺機能を使用するための関数などを生成します。

ここでは初期設定関数のみ作成することとして、「Device Initialization」にチェックをいれて「次へ」をクリックしてください。

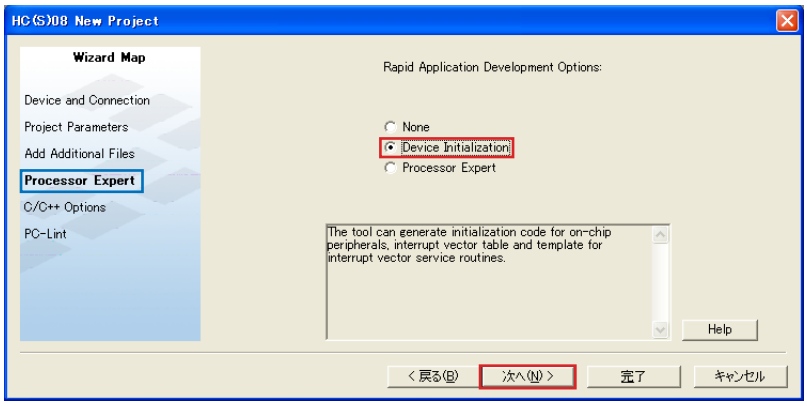


図 3-8 CodeWarrior：Processor Expert

「C/C++ Options」フェーズでは、作成するスタートアッププログラムの内容について設定します。以下のように設定して「完了」をクリックしてください。以上でプロジェクト作成のウィザードは終了です。

（Wizard Map の次のフェーズ「PC-Lint」は、Gimpel Software 社の構文解析ツール PC-Lint™（有償）を使用する際に必要な設定をおこないます）

- Which level of startup code…：「ANSI startup code」にチェックを入れます。

  「minimal startup code」→スタックポインタの設定を行います。

  「ANSI startup code」→スタックポインタの設定に加え、初期値の無い変数領域をゼロクリアし、初期値がある変数領域に初期値を設定します。

- Which memory model…：「Small」にチェックを入れます。

  「Tiny」→変数領域を 60h 番地から FFh 番地（ゼロページ）に配置します。スタック領域は 60h 番地から AFh 番地に配置され（80 バイト）、スタックポインタには AFh が設定されます。

  「Small」→変数領域を 100h 番地から配置します。スタック領域は 100h 番地から 14Fh 番地に配置され（80 バイト）、スタックポインタには 14Fh が設定されます。

- Select the froating point format…：「None」にチェックを入れます。

  「None」→浮動小数点を扱いません。コードサイズは他の 2 つの設定より抑えられます。

  「float is IEEE32, double is IEEE32」→ IEEE32 方式の float、double が扱えます。

  「float is IEEE32, double is IEEE64」→ IEEE32 方式の float、IEEE64 方式の double が扱えます。

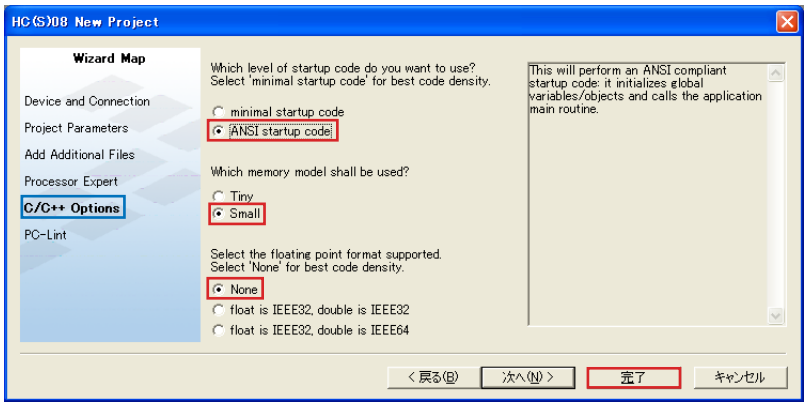

HC(S)08 New Project
Wizard Map
Device and Connection
Project Parameters
Add Additional Files
Processor Expert
C/C++ Options
PC-Lint
Which level of startup code do you want to use?
Select 'minimal startup code' for best code density.
minimal startup code
ANSI startup code
Which memory model shall be used?
Tiny
Small
Select the floating point format supported.
Select 'None' for best code density.
None
float is IEEE32, double is IEEE32
float is IEEE32, double is IEEE64
This will perform an ANSI compliant startup code: it initializes global variables/objects and calls the application main routine.
< 戻る(B)
次へ(N) >
完了
キャンセル

図3-9 CodeWarrior：C/C++ Options

プロジェクトウィザードが終わると、設定に従って各ファイルが生成され、「Project」ウィンドウと「Device Initialization」ウィンドウが開きます。（「Project Messages」、「Tips of the Day」ウィンドウは閉じてください。）

※　「Project Messages」に「Unable to replace macro…」というメッセージが表示されますが問題ありません。

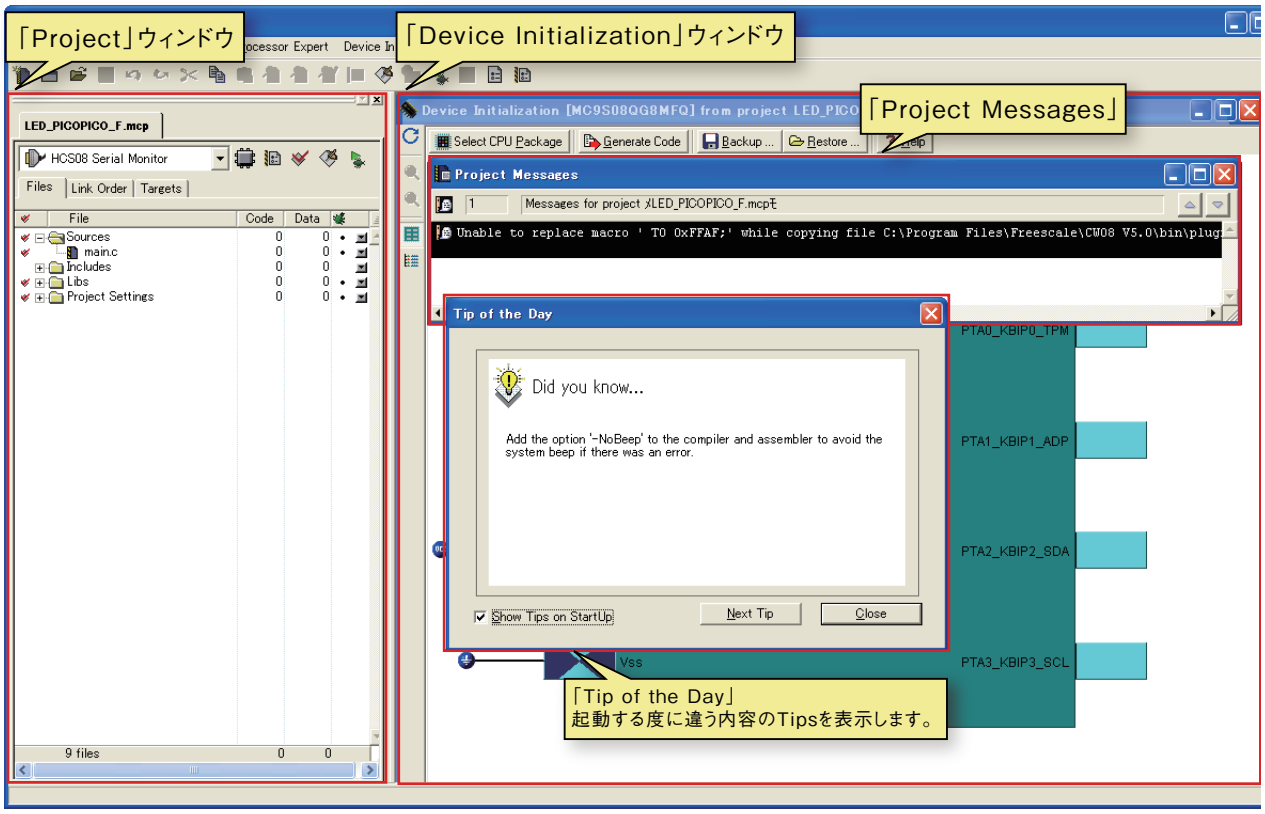


図3-10 CodeWarrior：初期画面

以下にプロジェクトウィザードで生成されたファイルを示します。

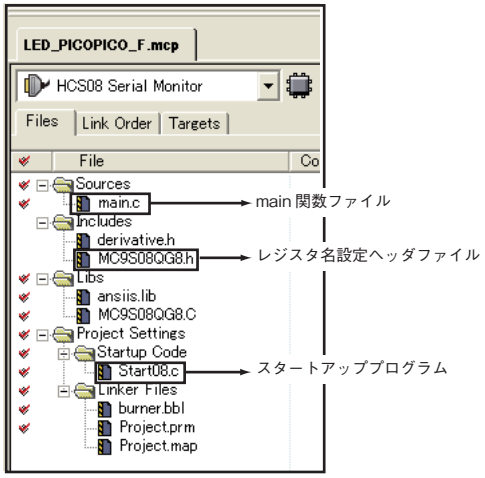


図3-11 CodeWarrior：プロジェクトウィザードで生成されたファイル

■日本語表示の設定

テキストエディタで日本語フォントが扱えるように設定します。「Edit」ウィンドウの「Preference...」をクリックしてください。「IDE Preference Panels」の「Editor」＞「Fonts & Tabs」をクリックします。

「Font」を日本語フォントに、「Script」を「日本語」に設定してください。設定が終わったら「OK」をクリックしてください。

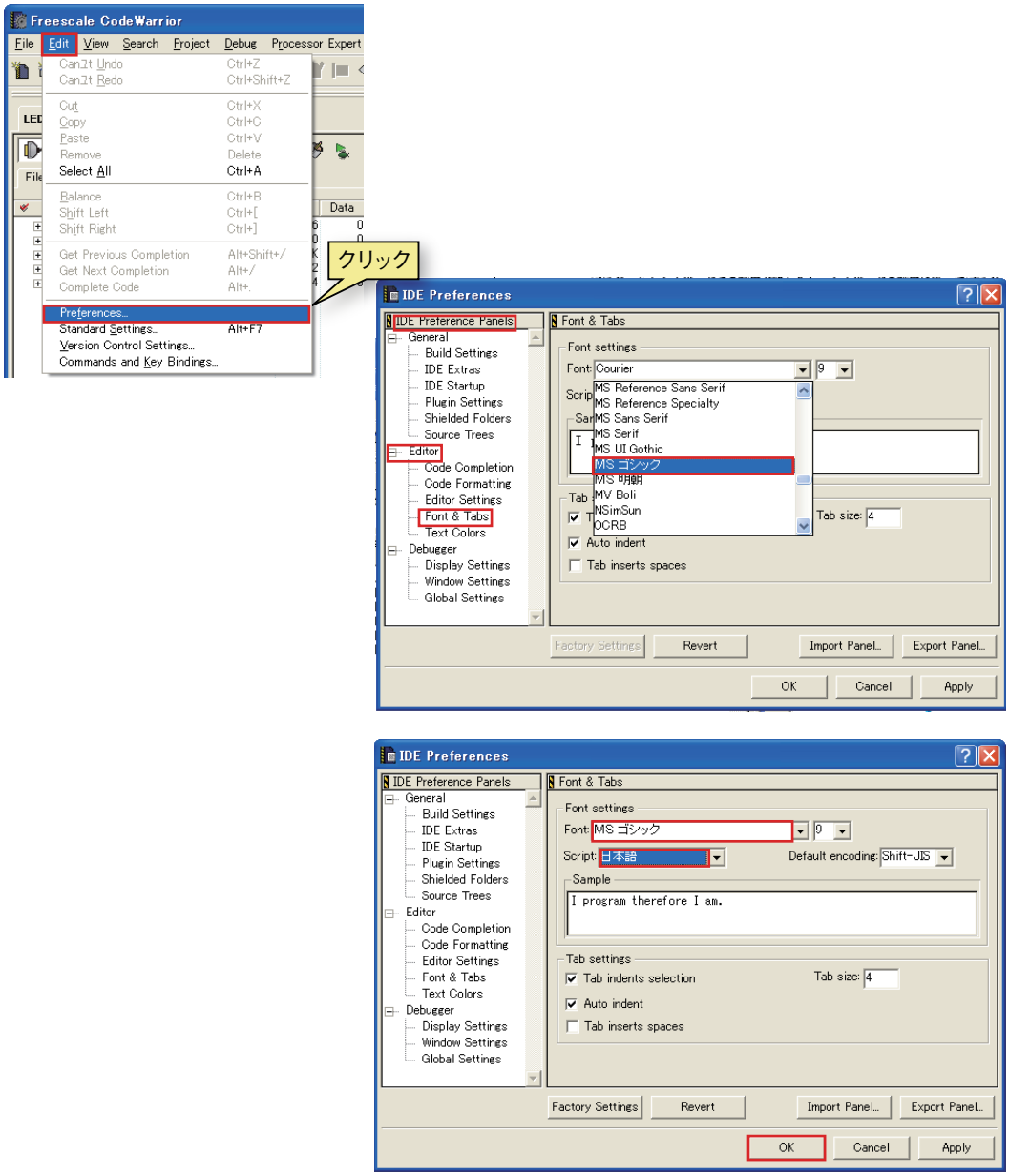


図3-12 CodeWarrior：日本語表示の設定

3.3 Device Initialization で初期設定関数を生成する

CodeWarrior では Device Initialization を使って、CPU や周辺機能の初期設定関数を GUI で簡単に生成できます。ここでは、CPU、PTB（PARALLEL INPUT/OUTPUT CONTROL MODULE）、TPM（16BIT TIMER/PWM MODULE）の初期設定関数を生成するための設定を行います。

■パッケージを設定する

「Device Initialization」ウィンドウの「Select CPU Package」をクリックして、「MC9S08QG8CPB」をクリックしてください。

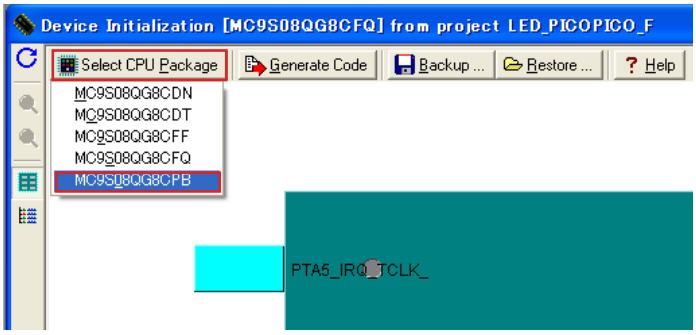


図3-13 CodeWarrior：Device Initializationのパッケージ設定

■CPUの設定をする

CPU の設定は、プログラム作成のたびに必ず行ってください。

「Device Initialization」ウィンドウの「CPU」ブロックをクリックしてください。

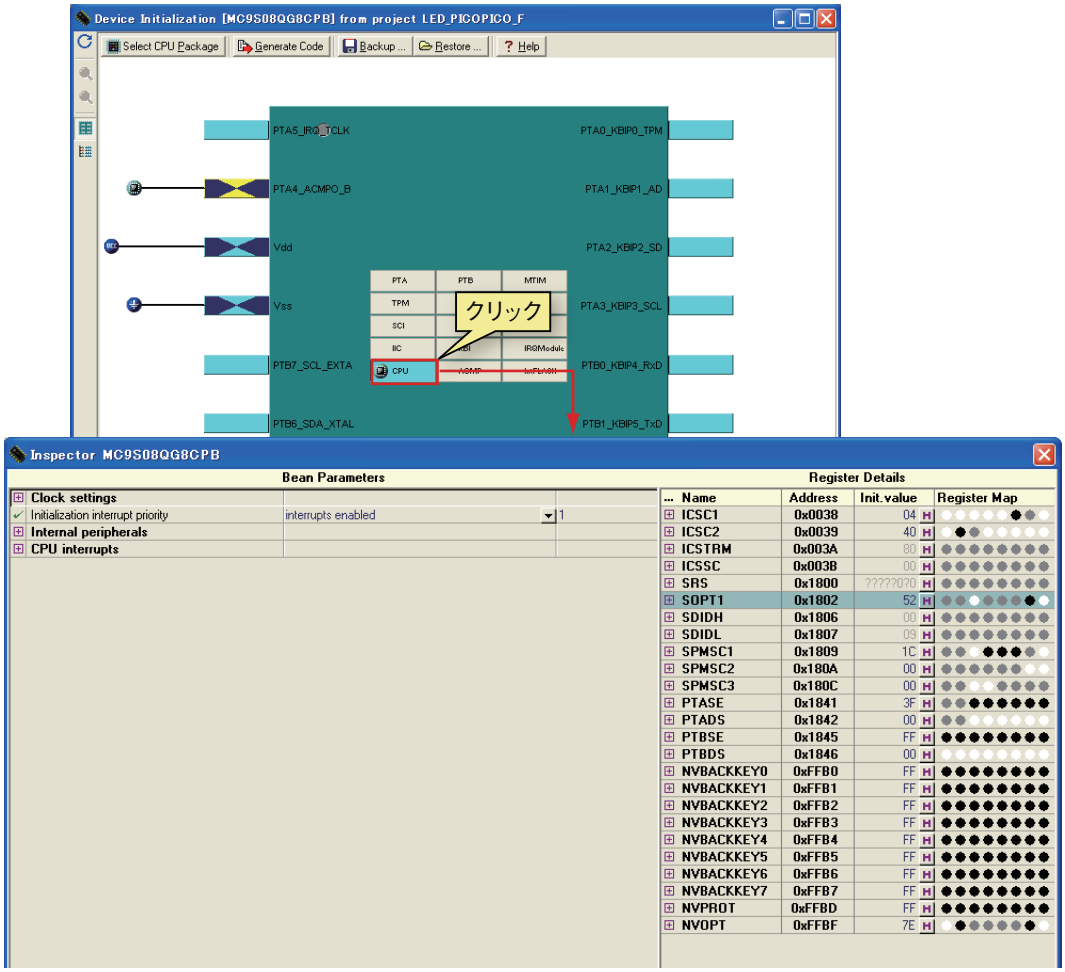


図3-14 CodeWarrior：CPUの設定

① バス・クロック 8MHz（BDIV による分周なし）に設定する

「Inspector MC9S08QG8CPB」ウィンドウの「Clock settings」左横の⊞ボタンをクリックしてください。設定レジスタが展開されます。

「Internal oscillator frequency[kHz]」を「31.25」に設定してください。

また、「Bus freq. divider」欄の「▼」をクリックし、プルダウンより「1」を選択してください。この設定により、「ICSC2」レジスタの設定値がデフォルトの「40」から「00」に変更されたことを確認してください。

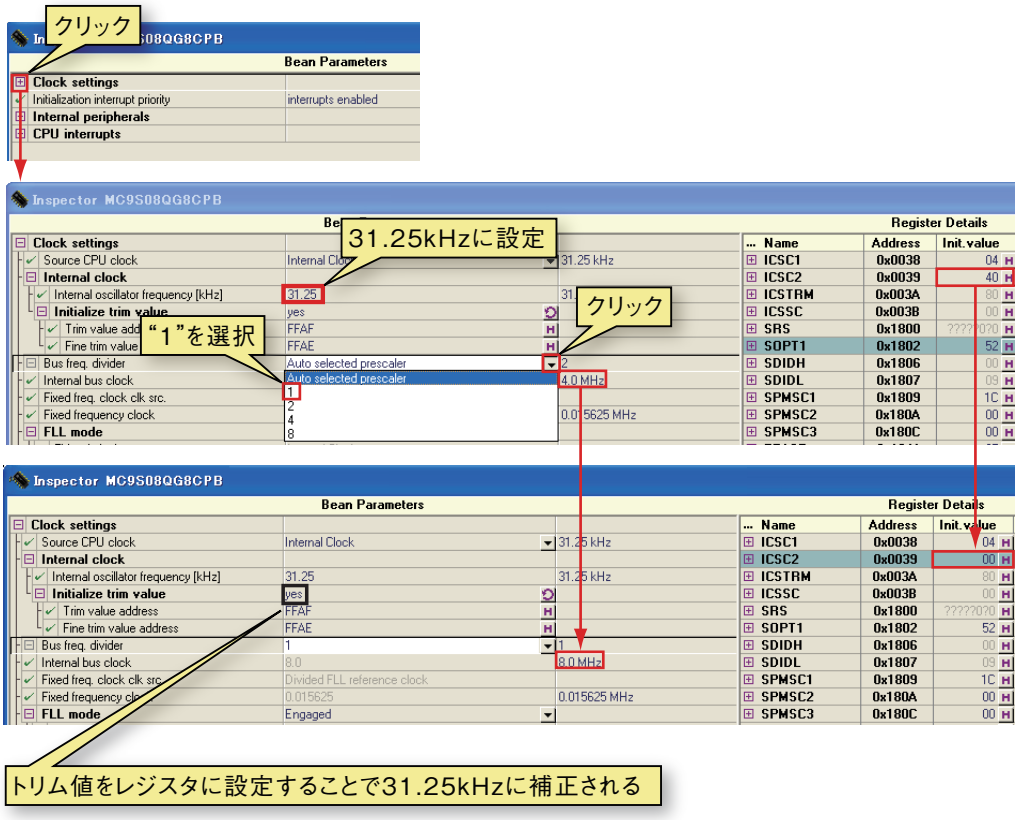


図3-15 CodeWarrior：バス周波数の設定

コラム：内部クロックについて

SF9S08C マイコンは、リセット解除後は FEI モード（FLL Engaged Internal）となり、マイコン内部で生成されるクロックによって動作します。内部クロックは標準値 32.7kHz（トリム値による補正なし Min. 25kHz ～ Max. 41.66kHz、動作温度約 20℃、3V 供給時）です。FEI モードでは、内部クロックは ICS ブロック（Internal Clock Source）内の FLL（Frequency Locked Loop：周波数同期回路）で 512 倍に逓倍され、ICS 制御レジスタ 2(ICSC2) の BDIV ビット（リセット解除後は 01=2 分周）の設定により分周されコア・クロックとして ICS ブロックより出力されます（ICSOUT）。コア・クロック（ICSOUT）はさらに 2 分周されてバス・クロック（BUSCLK）として供給されます。バス・クロックは、主な周辺機能や周辺機能の設定レジスタに供給されます。また、命令実行サイクルの基準クロックにもなっています。

SF9S08C マイコンには、個々のデバイスの内部クロック周波数のばらつきを補正するため、フラッシュメモリの特定のアドレスに内部クロックの周波数を 31.25kHz に補正するためのトリム値が、マイコン出荷時に書き込まれています。この値をフラッシュメモリから読み出して ICS トリムレジスタに設定することで、31.25kHz に補正することができます。

上記の初期設定では、トリム値を ICS トリムレジスタに設定することで 31.25kHz に補正し、さらに ICS 制御レジスタ 2(ICSC2) の BDIV ビット（Bus freq divider）を 00 = 1 分周に設定しているので、バスクロックは最終的に 8MHz となります。

バス・クロック = 内部クロック× 512 ÷（2 × BDIV ビットの設定※）

※　BDIV ビット：00 = 1 分周、01 = 2 分周、10 = 4 分周、11 = 8 分周

② PTA_IRQ_TCLK_RESET 端子をリセット端子に設定する

「Inspector MC9S08QG8CPB」ウィンドウの「Internal peripherals」左横の⊞ボタンをクリックしてください。設定レジスタが展開されます。「Reset pin support」欄の↻ボタンをクリックしてください。

この設定により、欄中央の表示が「Disabled」から「Enabled」に変わり、「SOPT1」レジスタがデフォルトの「52」から「53」に変更されたことを確認してください。以上で CPU の設定は終了です。「OK」をクリックしてください。

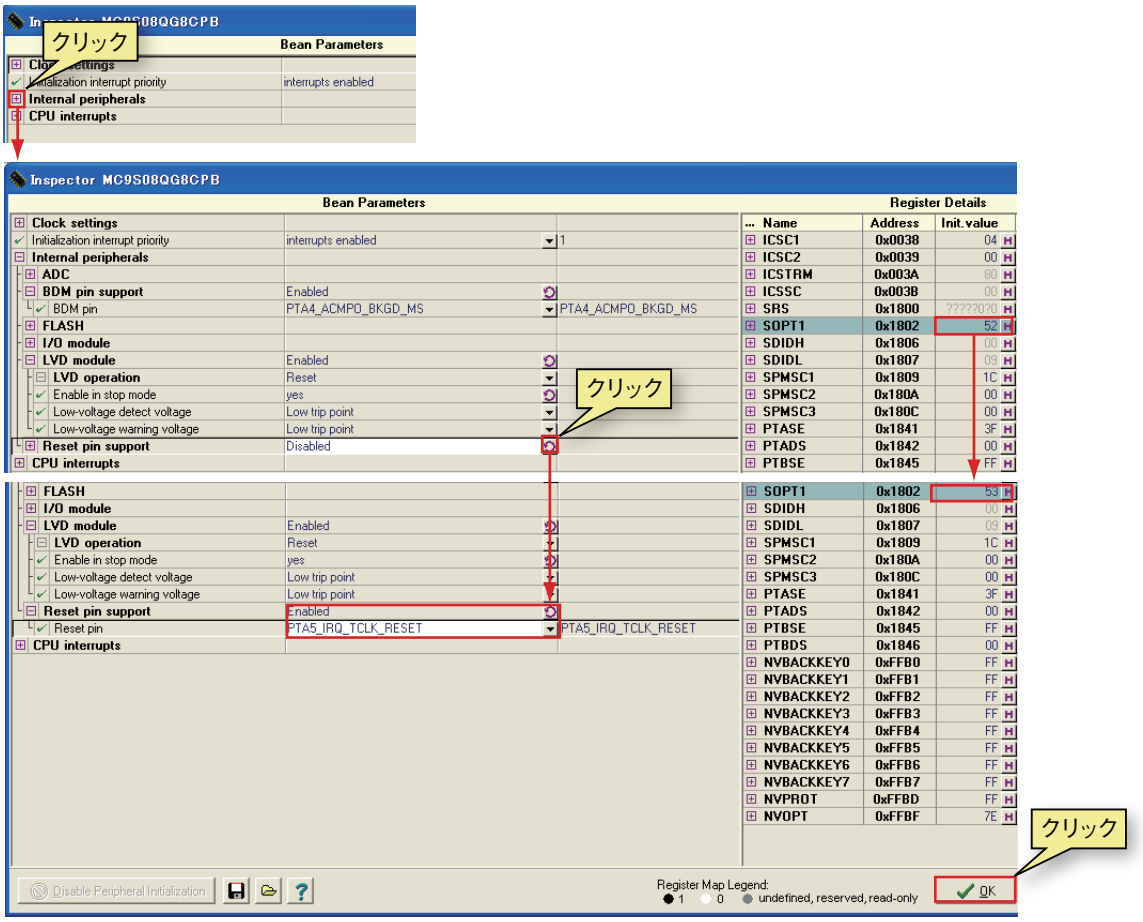


図3-16 CodeWarrior：リセット端子の設定

■PTBの設定をする

「Device Initialization」ウィンドウの「PTB」ブロックをクリックしてください

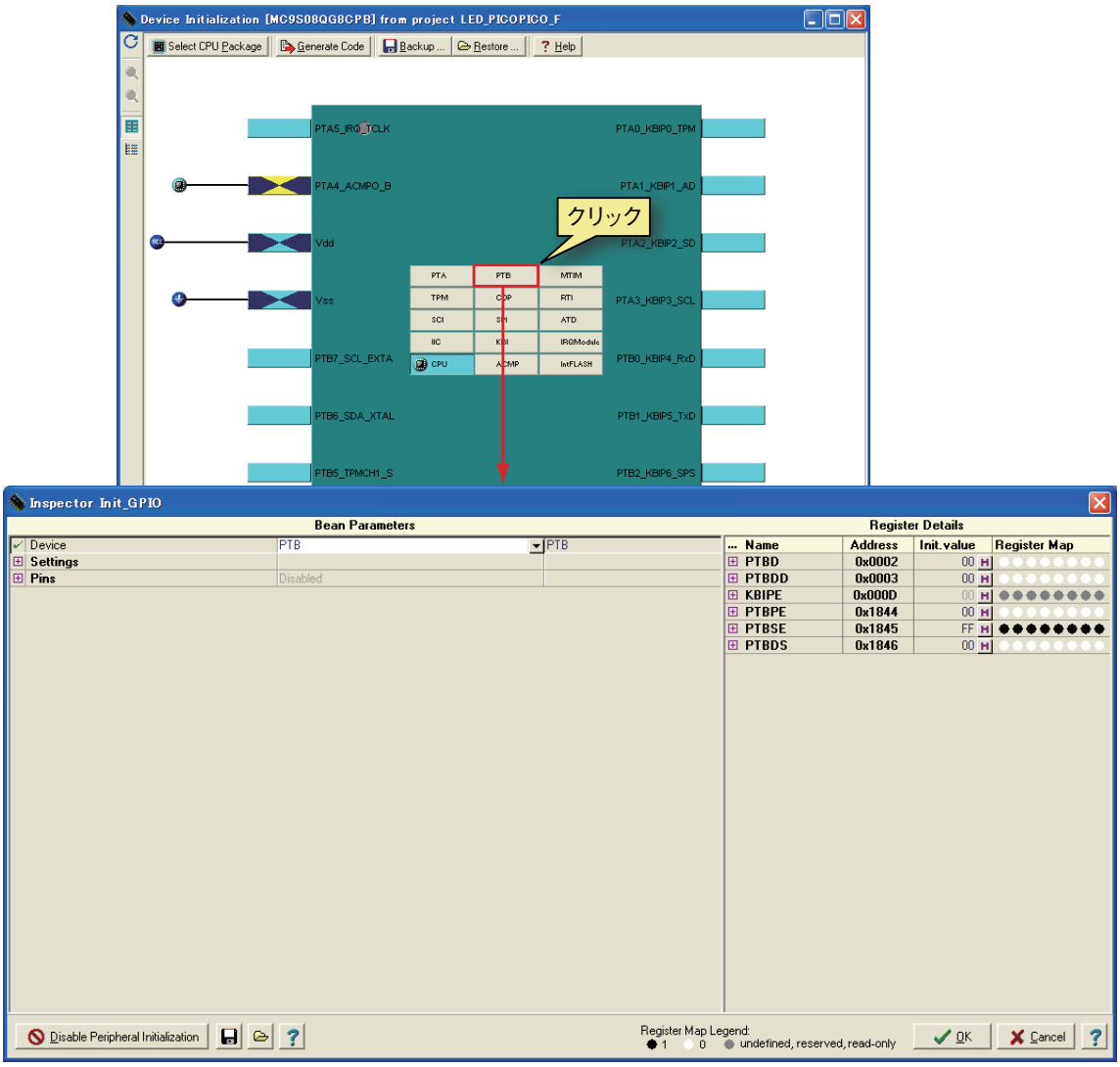


図3-17 CodeWarrior：PTB

① PTB6（LED1）を出力ポート、初期値“0”（点灯）に設定する

「Inspector Init_GPIO」ウィンドウの「Port Control」左横の ボタンをクリックしてください。ポートを端子ごとに設定できるようになります。

「Pin6」左横の ボタンをクリックして設定欄を展開し、「Pin6」右横の ボタンをクリックして「Enabled」にしてください。「Direction」をクリックし、プルダウンから「Output」を選択してください。また、「Output Value」欄に「0」を入力してください。

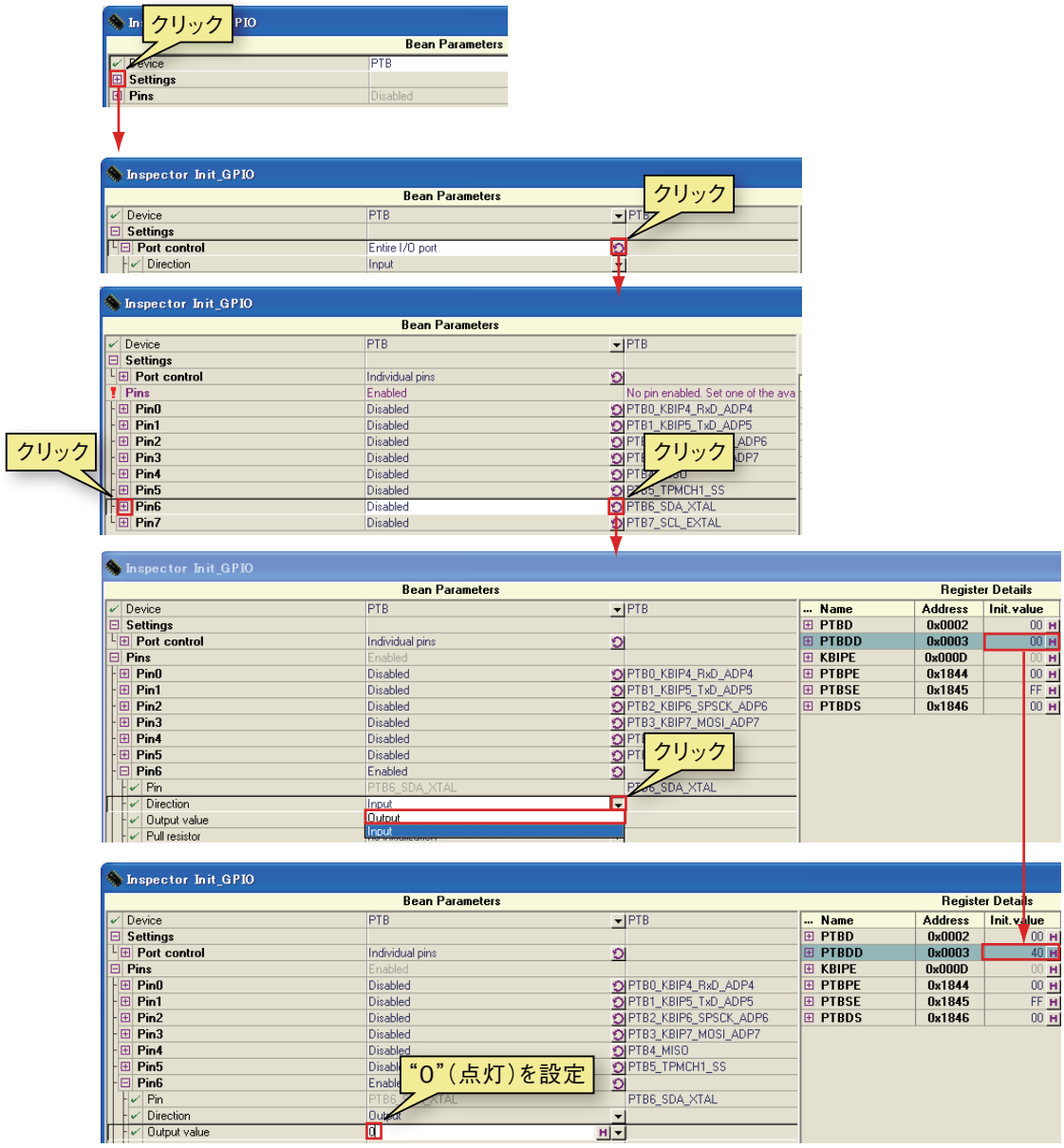


図3-18 CodeWarrior：PTB6の設定

② PTB7（LED2）を出力ポート、初期値“1”（消灯）に設定する

「Pin7」は「Pin6」と同じように展開して、「Direction」を「Output」に設定し、「Output Value」欄に「1」を入力してください。

①、②の設定で、「PTBD」レジスタが「80」に、「PTBDD」レジスタが「C0」に変更されているのを確認してください。以上で PTB の設定は終了です。「OK」をクリックしてください。

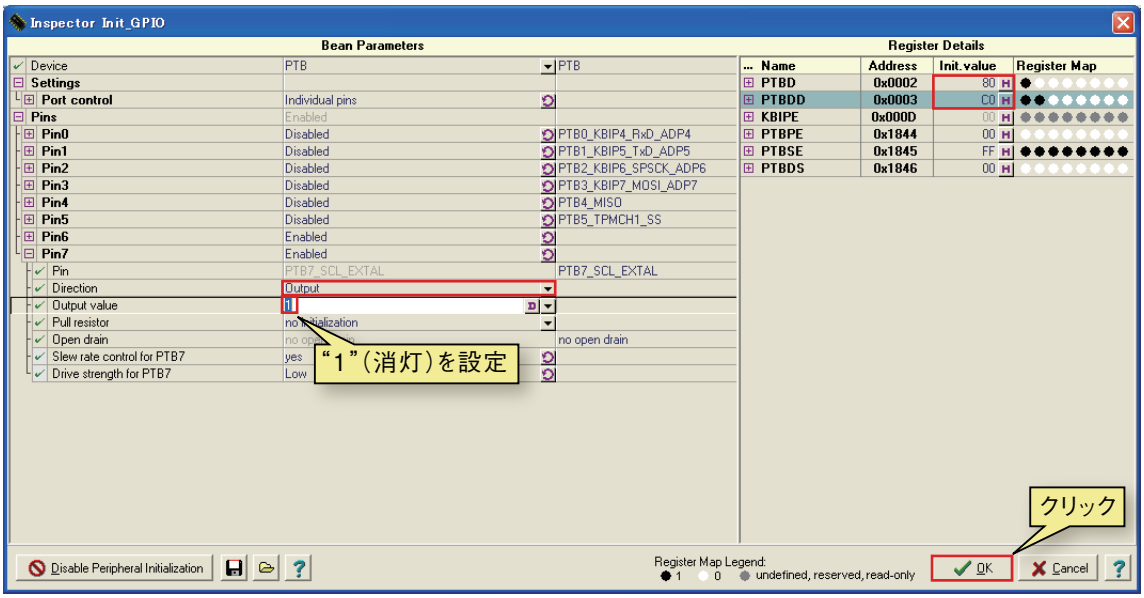


図3-19 CodeWarrior：PTB7の設定

■TPMの設定をする

「Device Initialization」ウィンドウの「TPM」ブロックをクリックしてください。

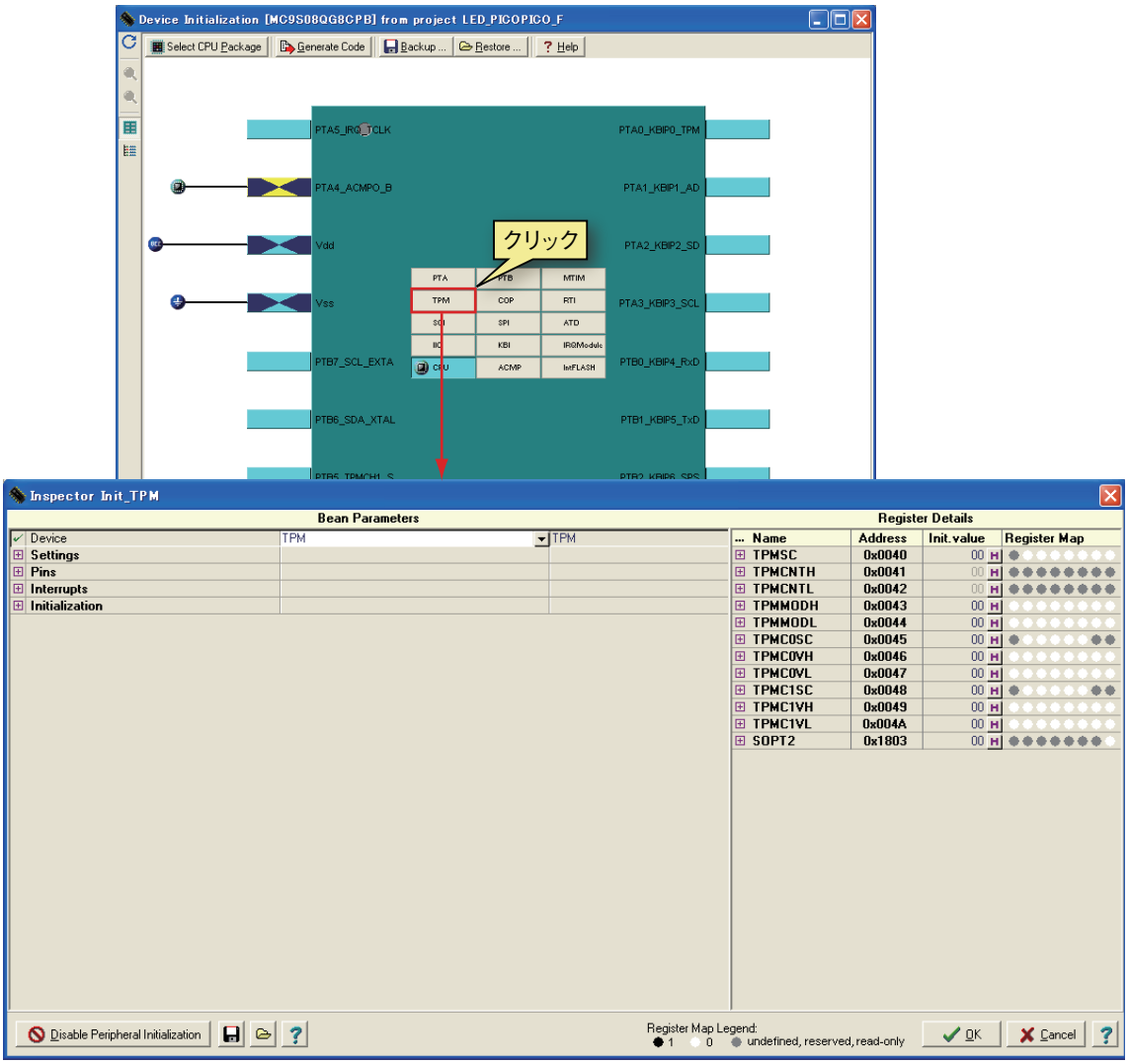


図3-20 CodeWarrior：TPMの設定

① TPM の供給クロックをバス周波数に設定する

「Inspector Init_TPM」ウィンドウの「Settings」左横の⊞ボタンをクリックしてください。設定レジスタが展開されます。「Clock Source Select」欄の「▼」をクリックしてください。プルダウンより「Bus rate clock」を選択してください。この設定により「TPMSC」レジスタの設定値が「0E」に変更されたことを確認してください。

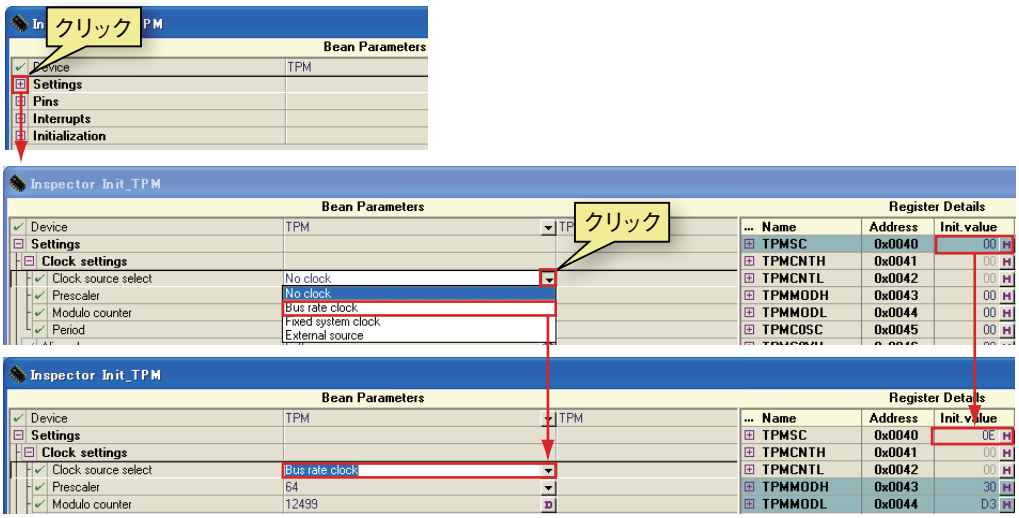


図3-21 CodeWarrior：TPM供給クロックの設定

コラム：TPM（Timer/Pulse-With Modulator）のカウント値について

SF9S08C マイコンの TPM は、供給クロックをプリスケーラで分周し、16 ビットのモジュロカウンタ（カウンタ値設定レジスタ）に設定した値をカウントする構成になっています。プリスケーラによる分周比は、1（分周なし）、2、4、8、16、32、64、128 より選択できます。LED 点滅プログラムの例では、供給クロックをバス・クロック（8MHz）にしているので、プリスケーラ分周比を 64、モジュロカウンタに 12500-1（0 もカウントされます）を設定して、0.1 秒（100ms）計測することとしています。

8MHz/64=125,000Hz（プリスケーラで分周されたクロック）　1 秒 /125,000Hz=0.000008 秒（クロックの周期）
0.1 秒 /0.000008 秒 =12500 回（0.1 秒計測するためのカウント数）

① プリスケーラ、モジュロカウンタの設定をする

0.1 秒（100m 秒）カウントするので、プリスケーラに「64」、モジュロカウンタに「12499（12500-1）」（それぞれ 10 進数）を設定することとします。

「Prescaler」欄の「▼」をクリックし、プルダウンより「64」を選択してください。この設定により、「TPMSC」レジスタの設定値が「08」から「0E」に変更されたことを確認してください。

「Modulo Counter」の数値入力欄横の「Radix Button」が「D」（10 進数）になっていることを確認して「12499」と入力してください。この設定により「TPMMODH」レジスタが「30」に、「TPMMODL」レジスタが「D3」に変更されたことを確認してください。

以上で TPM の設定は終了です。「OK」をクリックしてください。

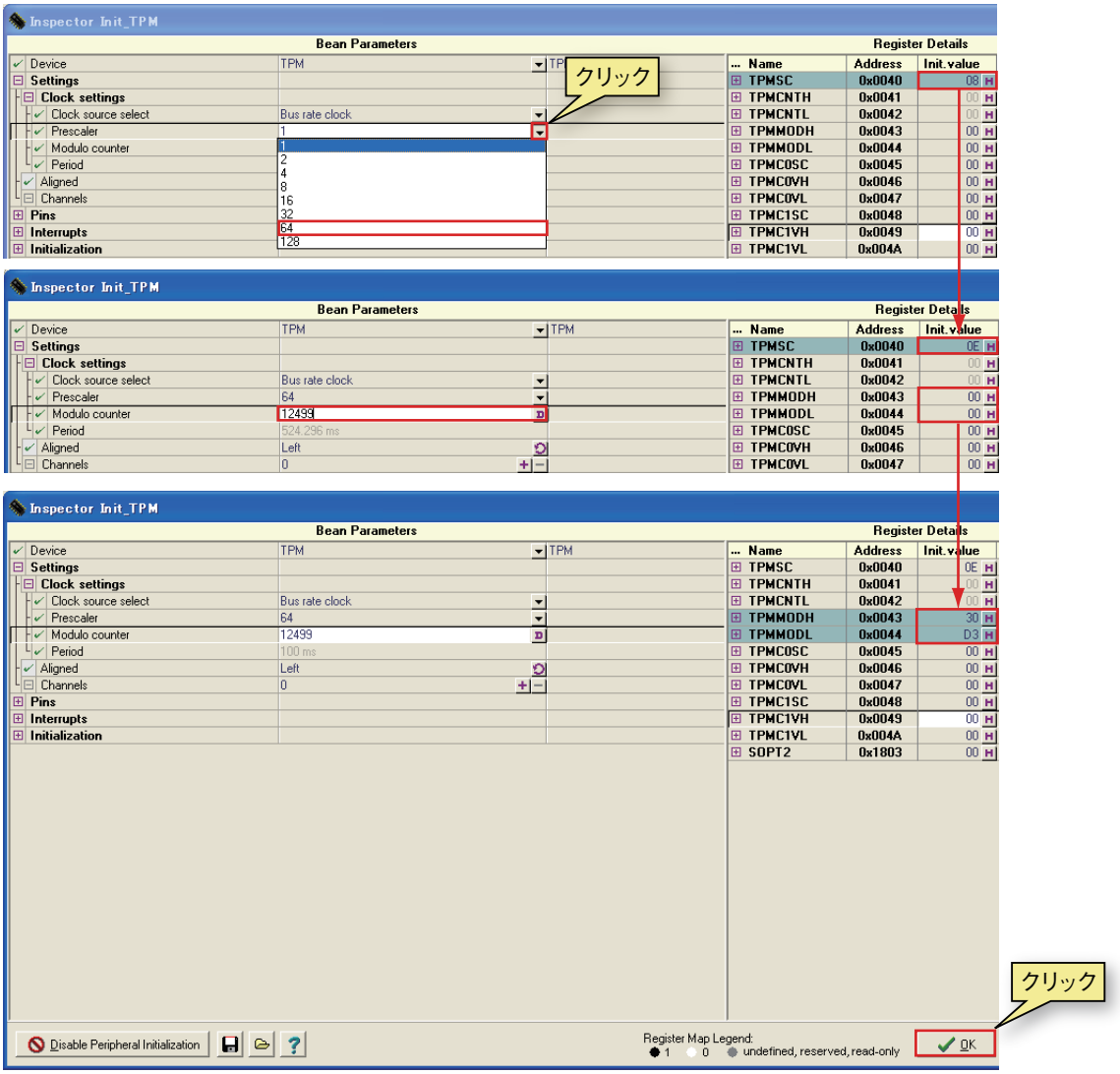


図3-22 CodeWarrior：プリスケーラ、モジュロカウンタの設定

■初期設定関数を生成する

これまでの設定を反映した関数を生成します。「Device Initialization」ウィンドウの「Generate Code」をクリックしてください。

「Device Initialization Options」ウィンドウが開きますが、設定はそのままで「Generate」をクリックしてください。初期設定関数が生成されます。

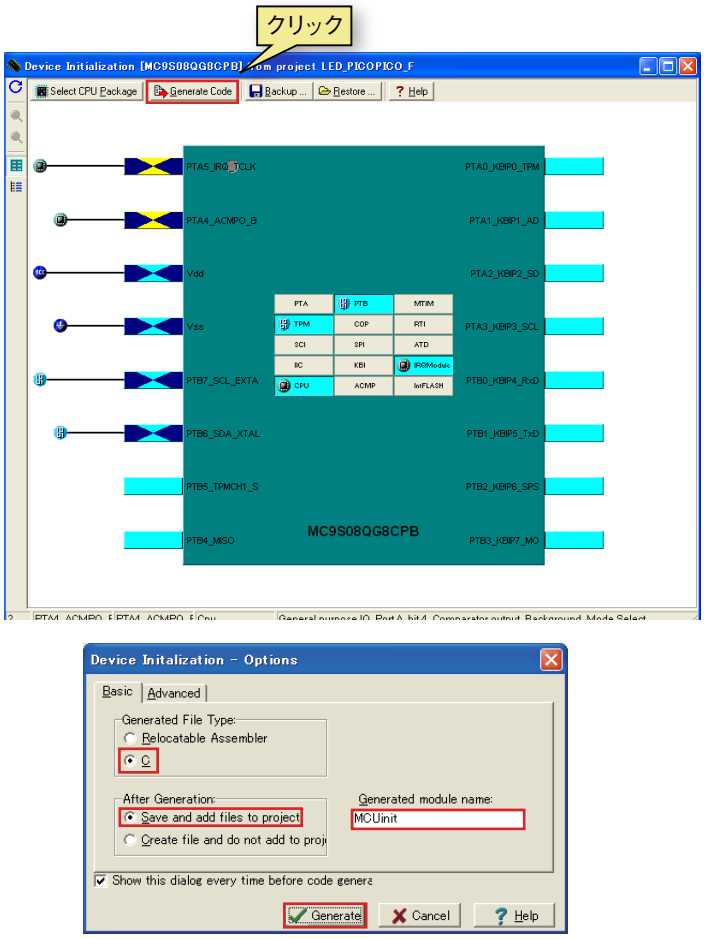


図3-23 CodeWarrior：Generate Code

初期設定関数の生成が終了すると「How to use…」ウィンドウが開きます。「OK」をクリックして閉じてください。

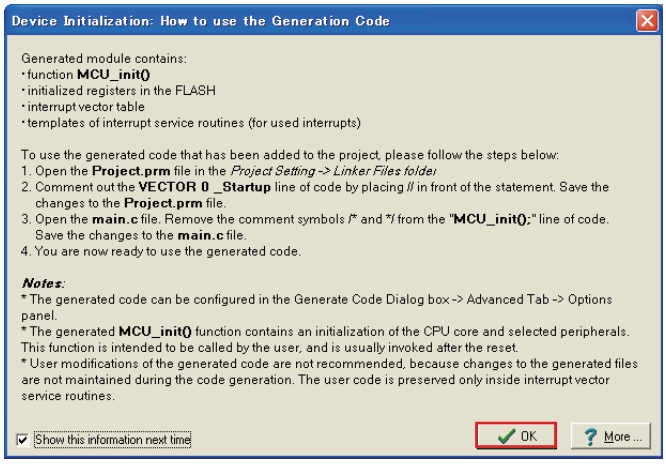

Device Initialization: How to use the Generation Code
Generated module contains:
• function MCU_init()
• initialized registers in the FLASH
• interrupt vector table
• templates of interrupt service routines (for used interrupts)
To use the generated code that has been added to the project, please follow the steps below:
1. Open the Project.prm file in the Project Setting -> Linker Files folder
2. Comment out the VECTOR 0 _Startup line of code by placing // in front of the statement. Save the changes to the Project.prm file.
3. Open the main.c file. Remove the comment symbols /* and */ from the "MCU_init();" line of code. Save the changes to the main.c file.
4. You are now ready to use the generated code.
Notes:
* The generated code can be configured in the Generate Code Dialog box -> Advanced Tab -> Options panel.
* The generated MCU_init() function contains an initialization of the CPU core and selected peripherals. This function is intended to be called by the user, and is usually invoked after the reset.
* User modifications of the generated code are not recommended, because changes to the generated files are not maintained during the code generation. The user code is preserved only inside interrupt vector service routines.
Show this information next time
OK
More ...

図3-24 CodeWarrior：How to use…

「Project」ウィンドウに生成されたファイルが表示されます。

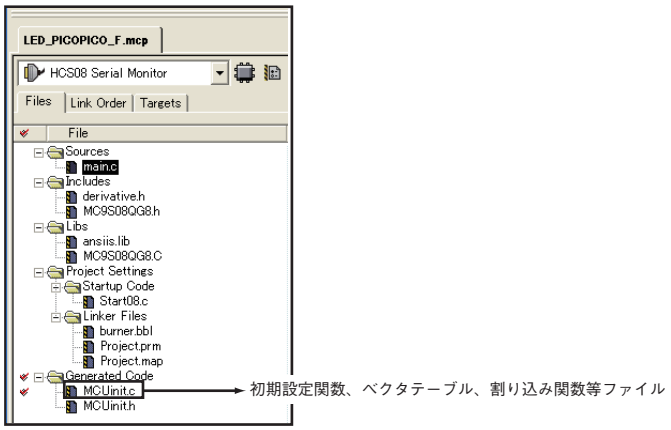


図3-25 Device Initializationで生成されたファイル

以下に各関数が実行される順を示します。

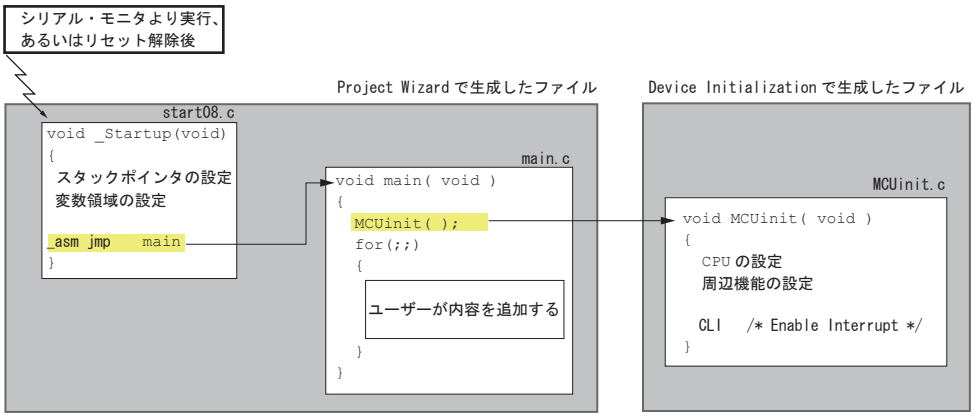


図3-26 各関数の実行順

3.4 シリアル・モニタに合わせて各ファイルの修正をする

「Device Initialization」により生成した各ファイルを、シリアル・モニタの制限事項に合わせて修正します。「Device Initialization」でファイルを生成した場合は、必ず行ってください。

■初期設定関数ファイル「MCUinit.c」を修正する

「Project」ウィンドウより「MCUinit.c」ファイルを開いてください。修正したら「保存」ボタン をクリックするか、キーボードより「Ctrl」＋「S」を押して保存してください。

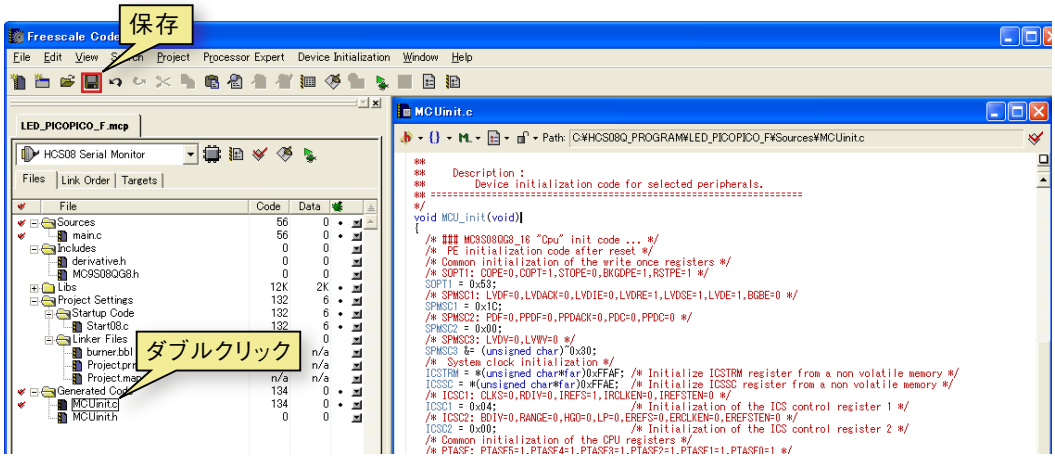


図3-27 CodeWarrior：ファイルの編集と保存

① 割り込みベクタを修正する

SF9S08C マイコンの割り込みベクタは FFC0h ～ FFFFh 番地の領域ですが、評価カード CT-298 搭載の SF9S08C マイコンにはあらかじめ、FC00 ～ FFFFh 番地の領域にシリアル・モニタが書き込まれていますので、ユーザは使用できません。FBD0h ～ FBFFh 番地にユーザが使用するベクタを設定してください。

割り込みベクタの設定箇所を以下のように修正してください。

- ベクタの開始アドレスを FFD0h から FBD0h に変更する
- シリアル・モニタの使用する割り込みベクタ（No.16,15,14,1）には FFFFh を設定し、それ以外には _Startup（ユーザプログラムの初期設定関数開始アドレス）を設定する。

```
/* Interrupt vector table */
#define UNASSIGNED_ISR 0xFFFF   /* unassigned interrupt service routine */

void (* const _vect[])() @0xFFD0 = {   /* Interrupt vector table */
    UNASSIGNED_ISR,     /* Int.no. 23 Vrti (at FFD0) */
    UNASSIGNED_ISR,     /* Int.no. 22 Reserved2 (at FFD2)  */
    UNASSIGNED_ISR,     /* Int.no. 21 Reserved3 (at FFD4) */
    UNASSIGNED_ISR,     /* Int.no. 20 Vacmp (at FFD6) */
    UNASSIGNED_ISR,     /* Int.no. 19 Vadc (at FFD8) */
    UNASSIGNED_ISR,     /* Int.no. 18 Vkeyboard (at FFDA) */
    UNASSIGNED_ISR,     /* Int.no. 17 Viic (at FFDC) */
    UNASSIGNED_ISR,     /* Int.no. 16 Vscitx (at FFDE)  */
    UNASSIGNED_ISR,     /* Int.no. 15 Vscirx (at FFE0)  */
    UNASSIGNED_ISR,     /* Int.no. 14 Vscierr (at FFE2) */
    UNASSIGNED_ISR,     /* Int.no. 13 Vspi (at FFE4) */
    UNASSIGNED_ISR,     /* Int.no. 12 Vmtim (at FFE6)*/
    UNASSIGNED_ISR,     /* Int.no. 11 Reserved13 (at FFE8) */
    UNASSIGNED_ISR,     /* Int.no. 10 Reserved14 (at FFEA) */
    UNASSIGNED_ISR,     /* Int.no.  9 Reserved15 (at FFEC) */
    UNASSIGNED_ISR,     /* Int.no.  8 Reserved16 (at FFEE)  */
    isrVtpmovf,         /* Int.no.  7 Vtpmovf (at FFF0)  */
    UNASSIGNED_ISR,     /* Int.no.  6 Vtpmch1 (at FFF2)  */
    UNASSIGNED_ISR,     /* Int.no.  5 Vtpmch0 (at FFF4) */
    UNASSIGNED_ISR,     /* Int.no.  4 Reserved20 (at FFF6)  */
    UNASSIGNED_ISR,     /* Int.no.  3 Vlvd (at FFF8) */
    UNASSIGNED_ISR,     /* Int.no.  2 Virq (at FFFA) */
    UNASSIGNED_ISR,     /* Int.no.  1 Vswi (at FFFC) */
    _Startup            /* Int.no.  0 Vreset (at FFFE) */
};
```

TPM の割り込みは使用しません。

```
/* Interrupt vector table */
#define UNASSIGNED_ISR 0xFFFF   /* unassigned interrupt service routine */

void (* const _vect[])() @0xFBD0 = {        /* Interrupt vector table */
    _Startup .          /* Int.no. 23 Vrti (at FFD0) */
    _Startup .          /* Int.no. 22 Reserved2 (at FFD2) */
    _Startup .          /* Int.no. 21 Reserved3 (at FFD4) */
    _Startup .          /* Int.no. 20 Vacmp (at FFD6) */
    _Startup .          /* Int.no. 19 Vadc (at FFD8) */
    _Startup .          /* Int.no. 18 Vkeyboard (at FFDA) */
    _Startup .          /* Int.no. 17 Viic (at FFDC) */
    UNASSIGNED_ISR,     /* Int.no. 16 Vscitx (at FFDE) */
    UNASSIGNED_ISR,     /* Int.no. 15 Vscirx (at FFE0) */
    UNASSIGNED_ISR,     /* Int.no. 14 Vscierr (at FFE2) */
    _Startup .          /* Int.no. 13 Vspi (at FFE4) */
    _Startup .          /* Int.no. 12 Vmtim (at FFE6)  */
    _Startup .          /* Int.no. 11 Reserved13 (at FFE8) */
    _Startup .          /* Int.no. 10 Reserved14 (at FFEA) */
    _Startup .          /* Int.no.  9 Reserved15 (at FFEC) */
    _Startup .          /* Int.no.  8 Reserved16 (at FFEE) */
    _Startup .          /* Int.no.  7 Vtpmovf (at FFF0)  */
    _Startup .          /* Int.no.  6 Vtpmch1 (at FFF2) */
    _Startup .          /* Int.no.  5 Vtpmch0 (at FFF4) */
    _Startup .          /* Int.no.  4 Reserved20 (at FFF6) */
    _Startup .          /* Int.no.  3 Vlvd (at FFF8) */
    _Startup .          /* Int.no.  2 Virq (at FFFA) */
    UNASSIGNED_ISR,     /* Int.no.  1 Vswi (at FFFC) */
    _Startup            /* Int.no.  0 Vreset (at FFFE) */
};
```

リスト3-1 割り込みベクタ（MCUinit.c）の修正

② NVPROT レジスタと NVOPT レジスタの設定を消去する

NVPROT レジスタと NVOPT レジスタはシリアル・モニタで設定済みですので、ユーザープログラムでは変更しないでください。下記に示す箇所を消去するかコメントアウトしてください。

```
/* Initialization of the CPU registers in FLASH */

/* NVPROT: FPS=0x7F,FPDIS=1    */
const volatile NVPROTSTR _NVPROT @0x0000FFBD = { 0xFF };

/* NVOPT: KEYEN=0,FNORED=1,SEC01=1,SEC00=0   */
const volatile NVOPTSTR _NVOPT @0x0000FFBF = { 0x7E };
```

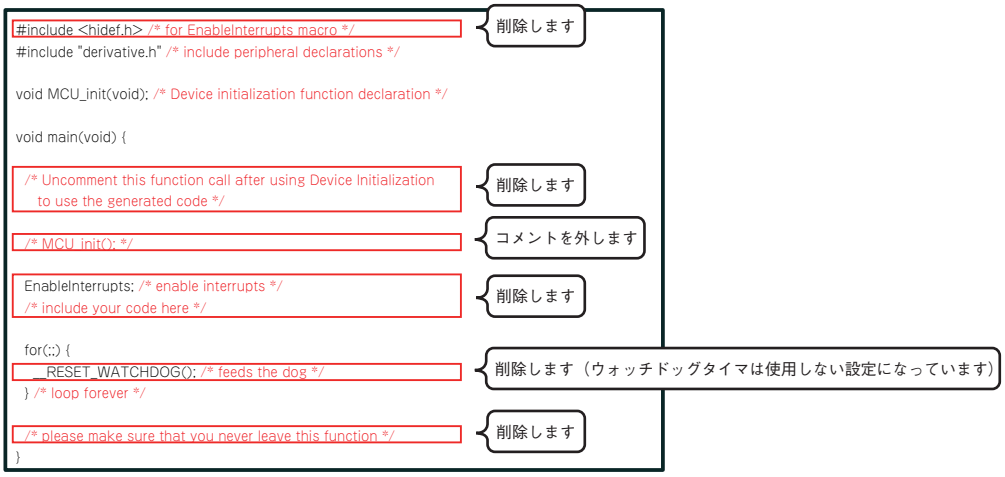
```
/* Initialization of the CPU registers in FLASH */

/* NVPROT: FPS=0x7F,FPDIS=1    */
/* const volatile NVPROTSTR _NVPROT @0x0000FFBD = { 0xFF }; */

/* NVOPT: KEYEN=0,FNORED=1,SEC01=1,SEC00=0   */
/* const volatile NVOPTSTR _NVOPT @0x0000FFBF = { 0x7E };   */
```

リスト3-2 レジスタ設定（MCUinit.c）の修正

3.5 プログラムの編集をする

「Project」ウィンドウより「main.c」ファイルを開いてください。

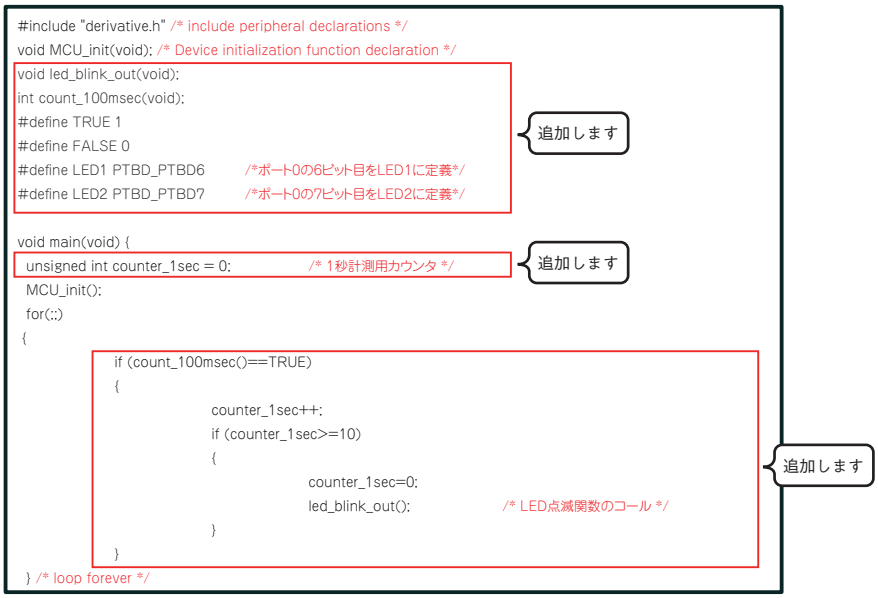
■必要のないコード、コメントを削除する



リスト3-4 コメント、コードの削除

■main関数をつくる

プロトタイプ宣言、定数の定義を記述し、main 関数の無限ループ内にタイマ計測、LED 出力関数のコールを追加します。



リスト3-5 時間計測、LED出力関数のコールの記述

■1秒計測関数、LED出力関数をつくる

1 秒計測関数、LED 出力関数を記述します。

```
#include "derivative.h"       /* include peripheral declarations */

void MCU_init(void);          /* Device initialization function declaration */
void led_blink_out(void);
int count_100msec(void);

#define TRUE 1
#define FALSE 0
#define LED1 PTBD_PTBD6       /*ポート0の6ビット目をLED1に定義*/
#define LED2 PTBD_PTBD7       /*ポート0の7ビット目をLED2に定義*/

void main(void)
{
  unsigned int counter_1sec = 0;                    /* 1秒計測用カウンタ */
  MCU_init();

  for(;;)
  {

    if (count_100msec()==TRUE)
    {
                counter_1sec++;

                if (counter_1sec>=10)
                {
                        counter_1sec=0;
                        led_blink_out();            /* LED点滅関数のコール */
                }
    }
  }
}

/* 1秒カウント関数 */
int count_100msec(void)
{
                if (TPMSC_TOF==1)                   /* タイマオーバーフローがあれば100msec経過 */
                {
                        asm
                        {
                          lda TPMSC                 /* タイマオーバーフローフラグクリアのためのTPMSCレジスタ読み出し(アセンブラ) */
                        }
                        TPMSC_TOF=0;                /* タイマオーバーフローフラグクリア */
                        return TRUE;
                }
                return FALSE;
}

/* LED点滅関数 */
void led_blink_out(void)
{
                LED1 = ~LED1;                       /* LED1反転 */
                LED2 = ~LED2;                       /* LED2反転 */
}
```

追加します

リスト3-6 1秒計測関数、LED出力関数の記述

以上でプログラムの編集は終了です。「main.c」ファイルを保存してください。

コラム：TPMの割り込みについて

TPM の割り込みをポーリング（処理の要求を定期的に確認する方式のこと）する場合は、割り込みの要因となる TOF（タイマオーバーフローフラグ）を検知します。

また、TPM の TOF をクリアするには、TPMSC（TPM ステータスコントロールレジスタ）を読み出してから TOF に 0 を設定します。

3.6 プログラムをコンパイルする

「Project」ウィンドウの「Make」ボタン を押して、プログラムをコンパイルします。エラーメッセージが出なければ、コンパイルの成功です。

下図は命令文の最後に「;」が抜けていた場合のエラーメッセージです。

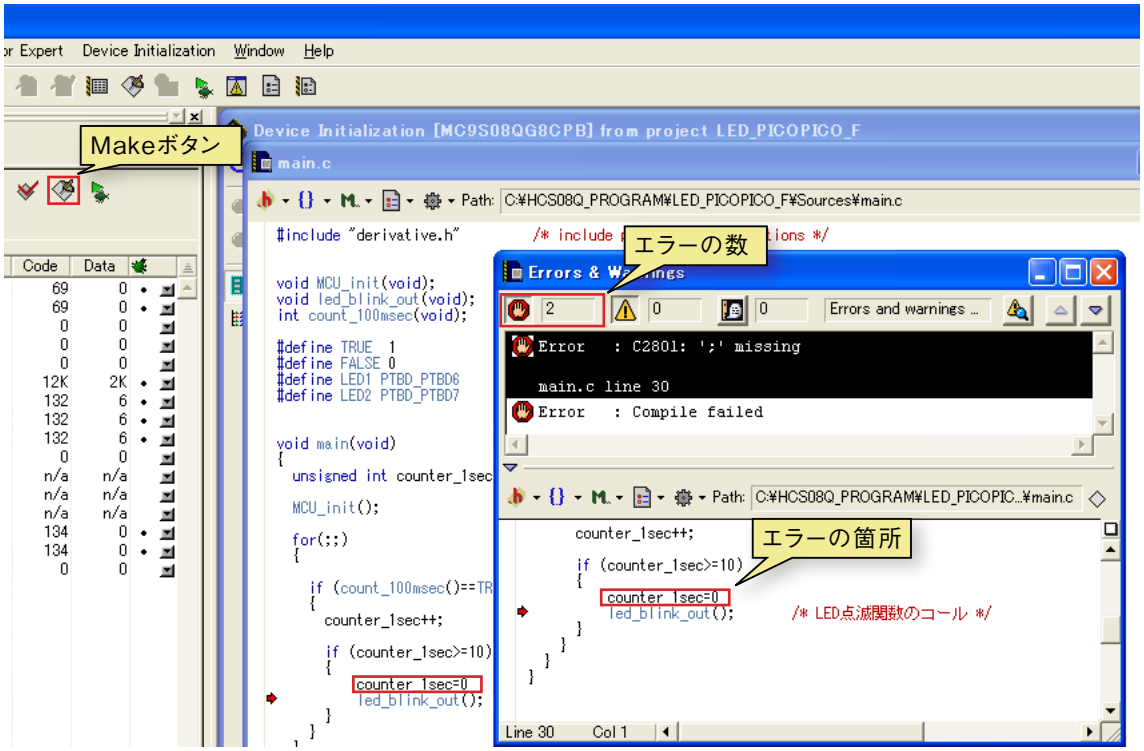


図3-28 CodeWarrior：「Make」ボタンと「Errors & Warnings」ウィンドウ

4. プログラムをうごかそう

CodeWarrior のデバッガでプログラムをマイコンにダウンロードし、動かしてみましょう。

4.1 CodeWarrior からデバッガを起動する

評価カード CT-298 の SW1 を押している状態で、ホスト PC と USB ケーブルで接続してください。

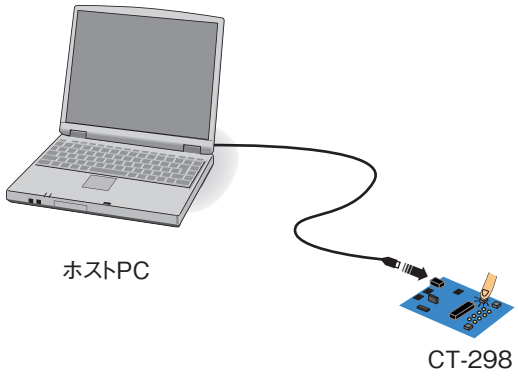


図4-1 CT-298とホストPCとの接続

「Project」ウィンドウの「Debug」ボタン をクリックしてください。デバッガが起動します。

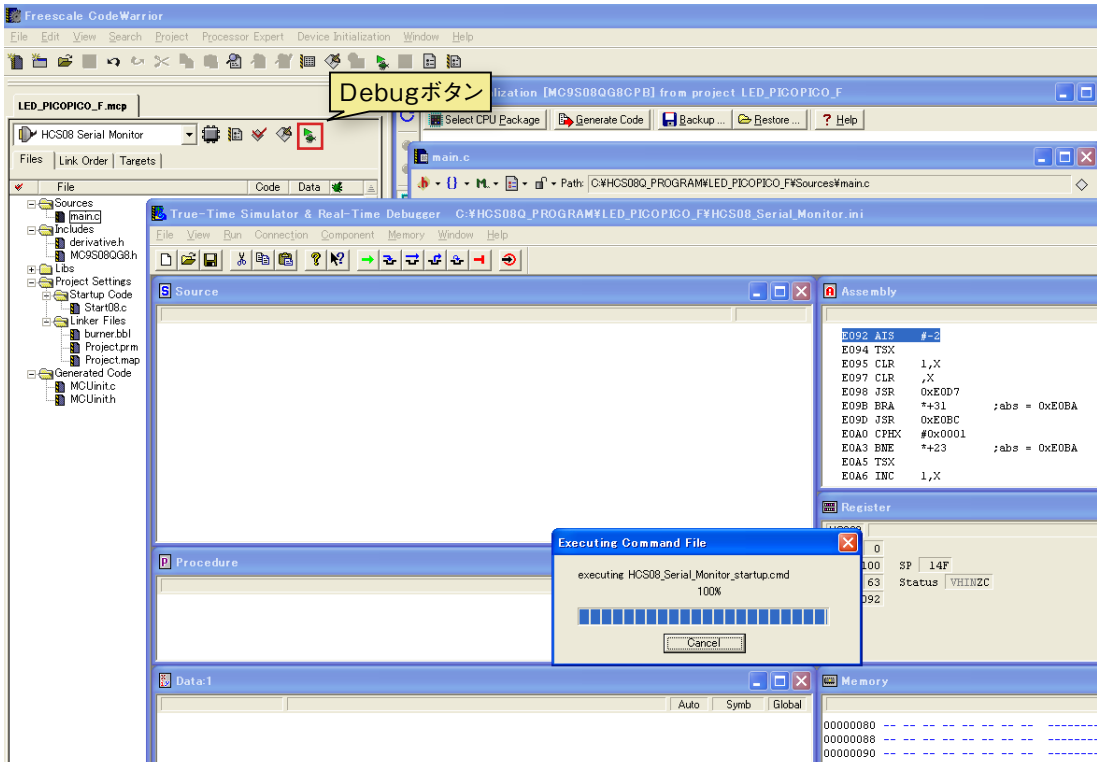


図4-2 CodeWarrior：デバッガとホストPCとの通信

「Set Derivative」の設定画面が出た場合、「MC9S08QG8」に設定されていることを確認して「OK」をクリックしてください。

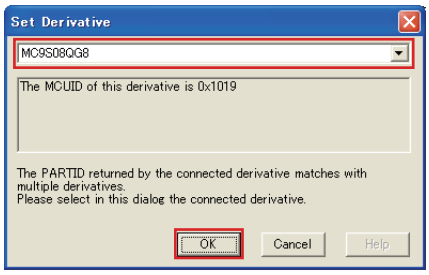

Set Derivative
MC9S08QG8
The MCUID of this derivative is 0x1019
The PARTID returned by the connected derivative matches with multiple derivatives.
Please select in this dialog the connected derivative.
OK
Cancel
Help

図4-3 CodeWarrior：デバッガ「Set Derivative」ウィンドウ

「Monitor Setup」の設定画面が出た場合、「Enable Vector Mirroring」にチェックを入れ、「Auto Detect」ボタンをクリックしてください。右側のボックスに数値が入ったことを確認して「OK」をクリックしてください。

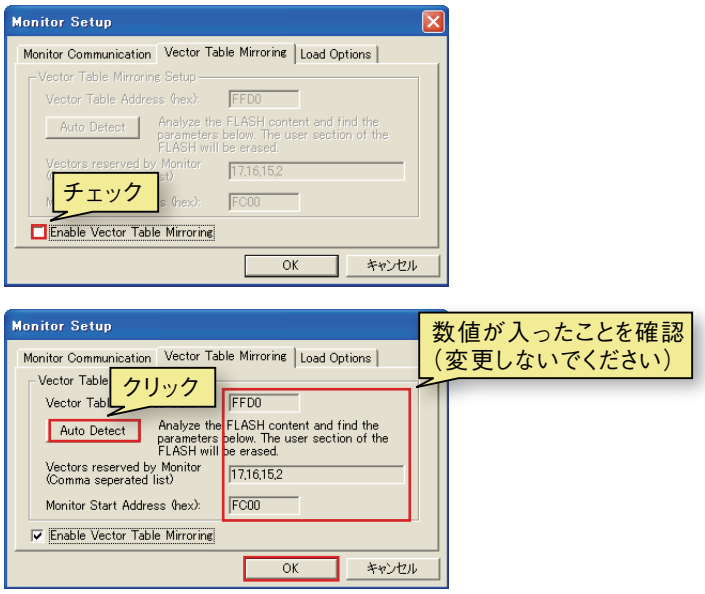


図4-4 CodeWarrior：デバッガ「Monitor Setup」ウィンドウ

注意

プログラムのダウンロード中にホスト PC、ターゲットシステムのパワーオフ、リセットはしないでください。

ホスト PC との通信が確立するとコンパイルしたオブジェクトファイルをダウンロードします。デバッガは、ダウンロードしたユーザープログラムのスタートアッププログラム（start08.c）を実行し、main 関数の先頭で止まった状態になります。

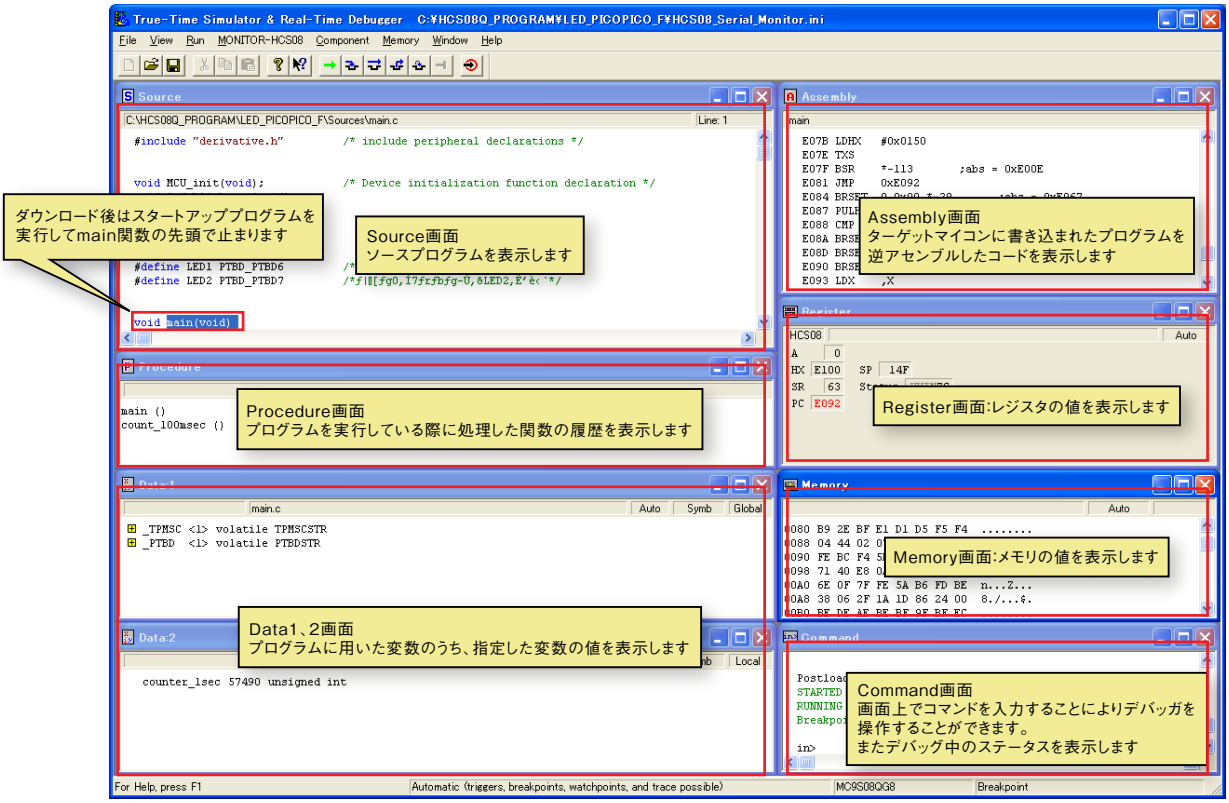


図4-5 CodeWarrior：デバッガウィンドウ

日本語が表示されるように設定します。「Component」メニューより「Fonts…」をクリックしてください。「フォント名」に日本語フォントを、「文字セット」に「日本語」を設定して「OK」をクリックしてください。

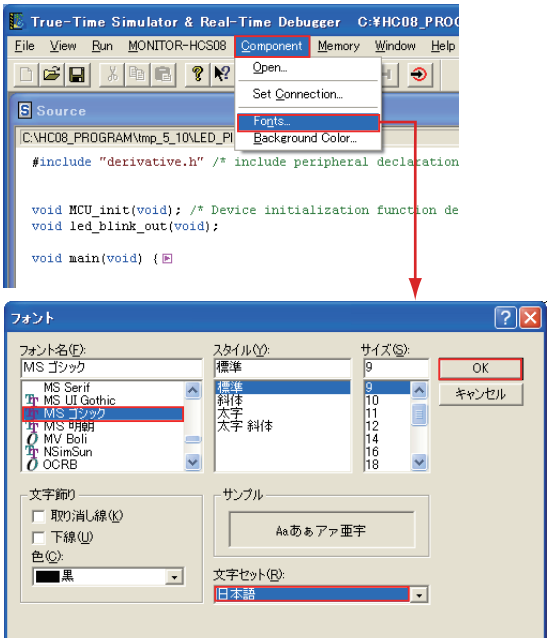


図4-6 CodeWarrior：デバッガの日本語設定

4.2 プログラムの実行と停止

「Start/Continue」ボタン をクリックすると、main 関数よりプログラムが実行されます。評価カード CT-298 上の 2 つの LED が交互に点滅するのを確認してください。

「Halt」ボタン をクリックすると停止します。

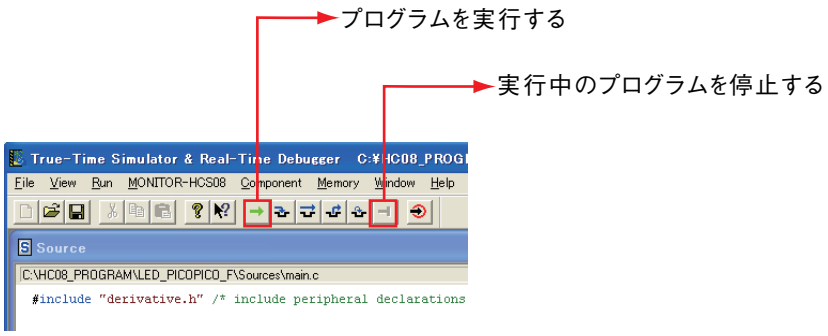


図4-7 CodeWarrior：プログラムの実行と停止

4.3 プログラムをステップ実行する

以下のボタンでステップ実行（1 命令づつ実行する）ができます。

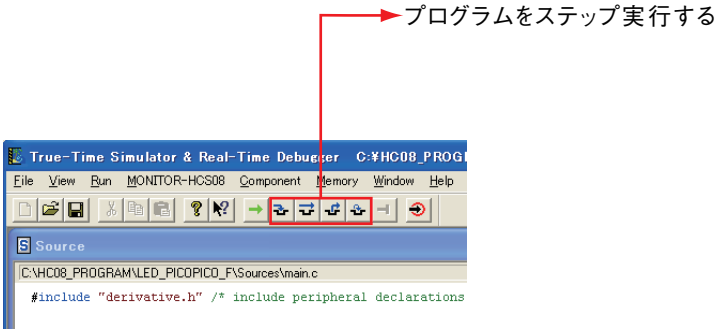


図4-8 CodeWarrior：プログラムのステップ実行

Single Step：プログラムを 1 行づつ実行する

Step Over：関数を 1 回実行し、戻ったところで停止する

Step Out：現在の関数から戻ったところで停止する

Assembly Step：アセンブリ言語レベルでプログラムを 1 行づつ実行する

4.4 プログラムの開始アドレスを変更する

開始したい行にマウスを置き、ダブルクリックで命令を選択します。そのままレジスタ画面の PC（プログラムカウンタ）へドラッグします。

あるいは命令が選択されている状態で右クリックし、ポップアップメニューの「Set Program Counter」をクリックします。

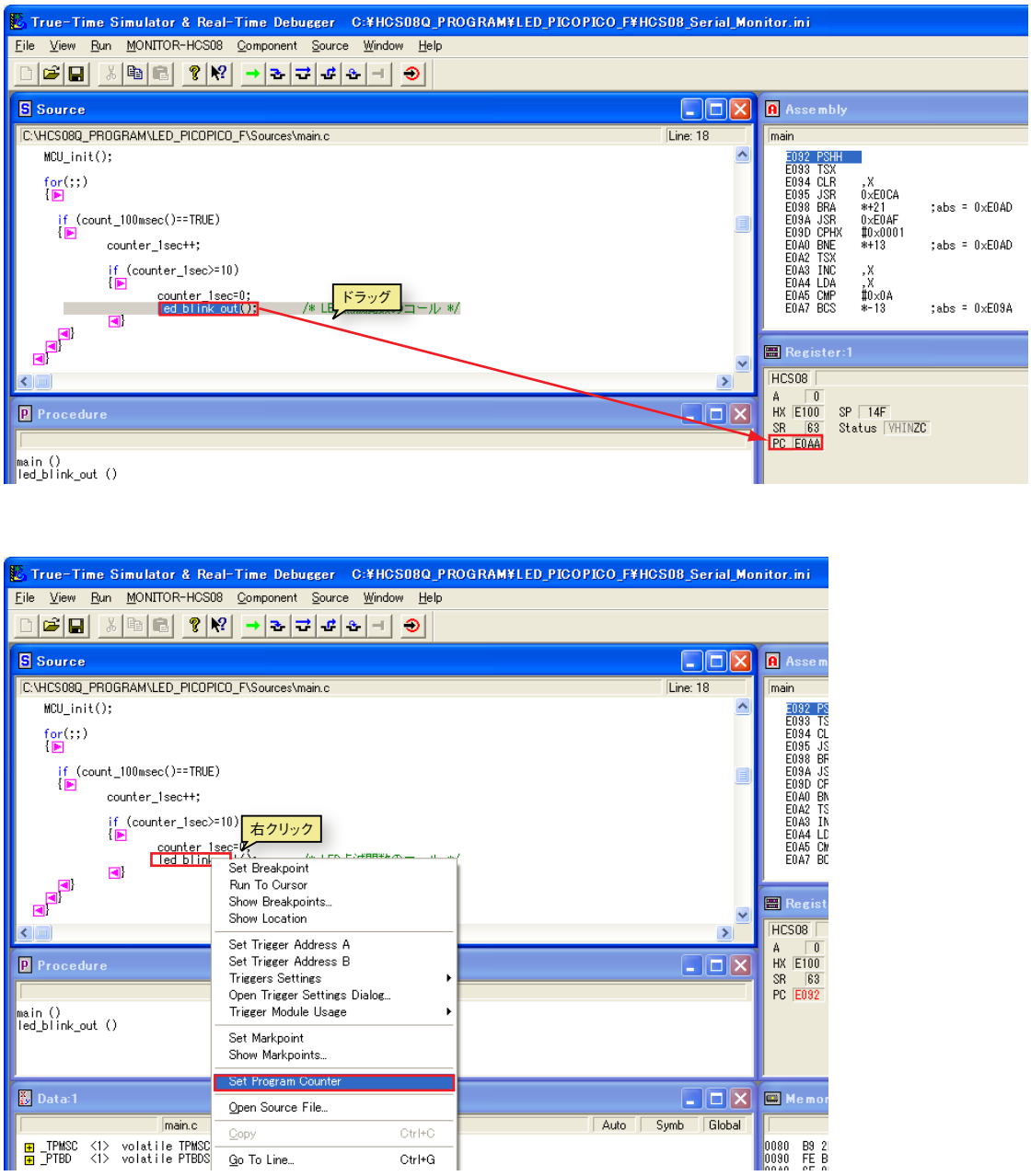


図4-9 CodeWarrior：プログラムの開始アドレスの変更

4.5 プログラムの停止条件を設定する

通常のブレークの他に、実行命令の順番やメモリのアクセスを条件にプログラムを停止させることができます。

■ブレークポイントを設定する

停止させたい行をダブルクリックし、右クリックでポップアップメニューを開き、「Set Breakpoint」をクリックしてください。ブレークポイントは2つまで設定できます。設定したブレークポイントの解除は、同じポップアップメニューより「Delete Breakpoint」をクリックしてください。

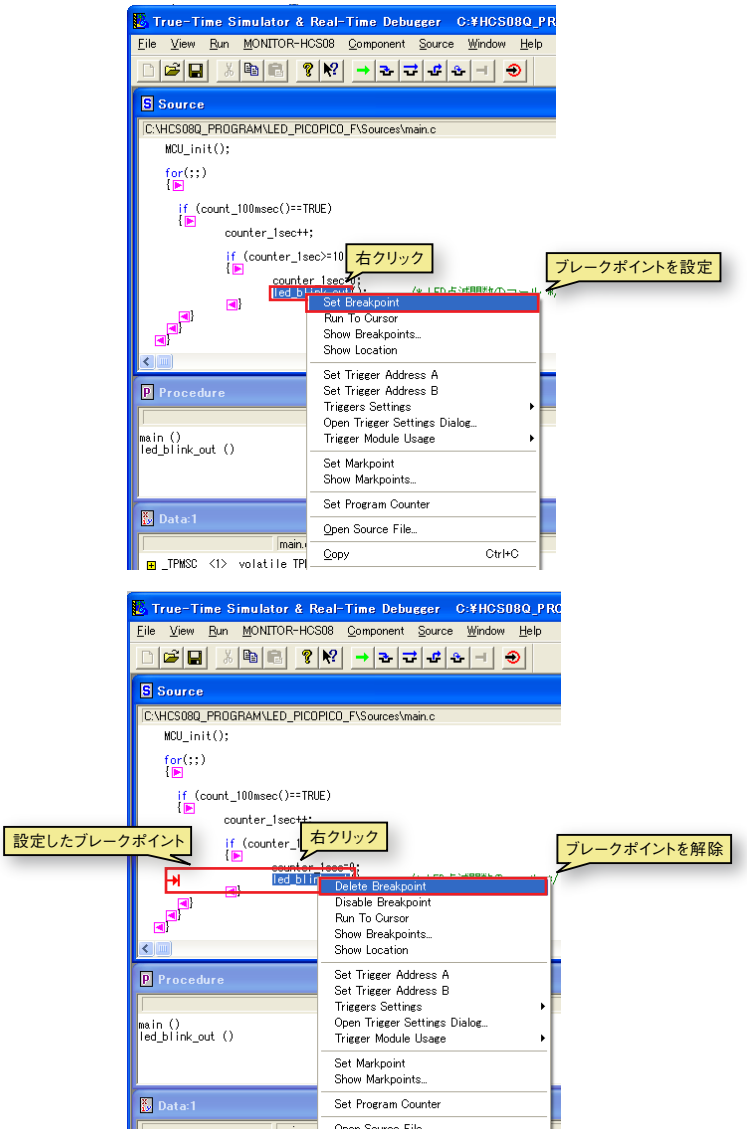


図4-10 CodeWarrior：ブレークポイントの設定と解除

■行にトリガを設定する

行（インストラクション）にトリガを設定し、停止する条件を設定できます。以下の例では、トリガＡの番地を実行してからトリガＢを実行した時点でプログラムが停止します。トリガはＡとＢの２つまで設定できます。ブレークとトリガを同時に設定することはできません。

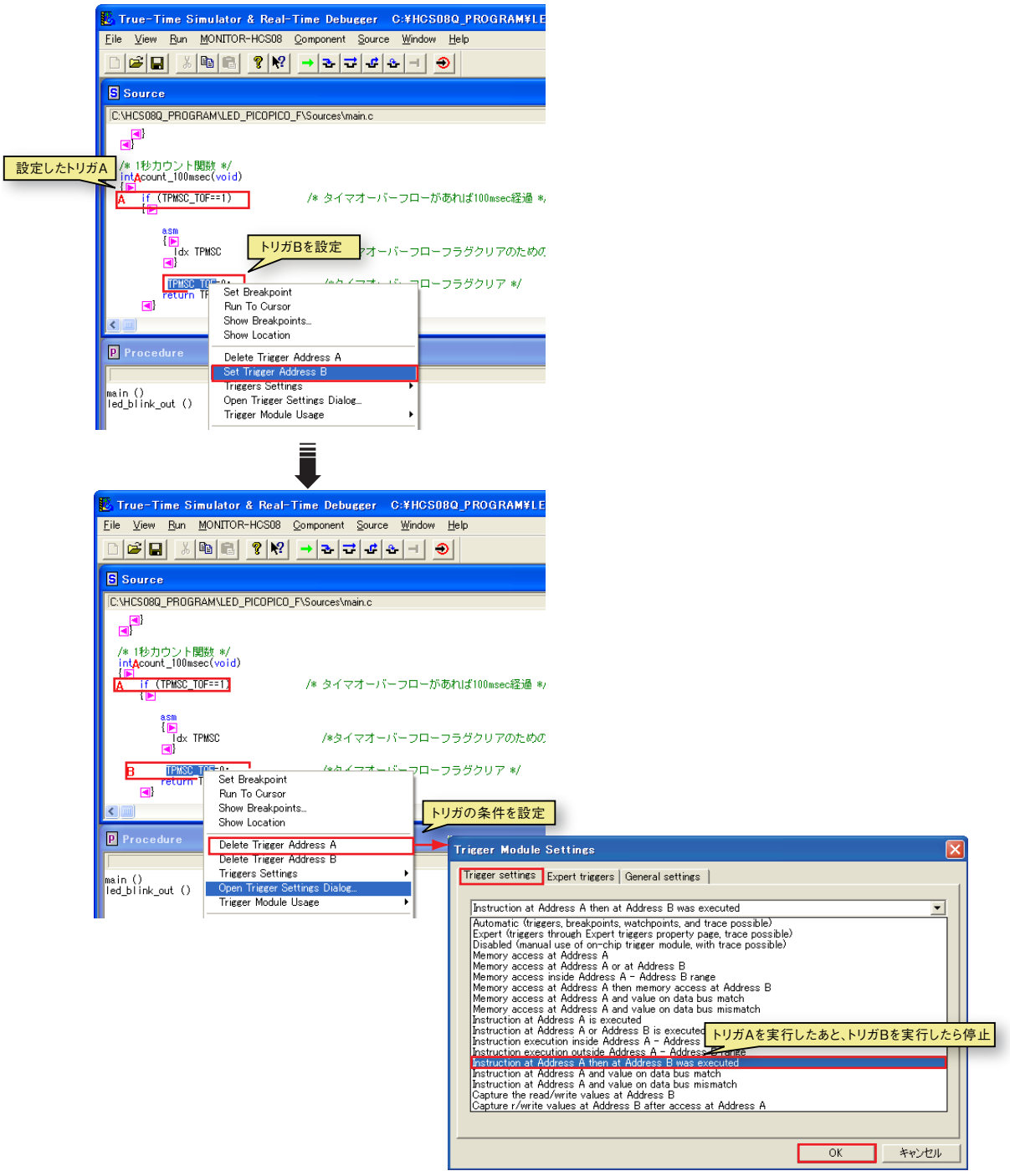


図4-11 CodeWarrior：トリガの設定

■メモリにトリガを設定する

メモリのアクセス（Read/Write あるいはどちらか１つ）を条件にプログラムを停止することができます。以下の例では、PTBD（ポート B のデータレジスタ）に書き込んだ時点でプログラムが停止します。

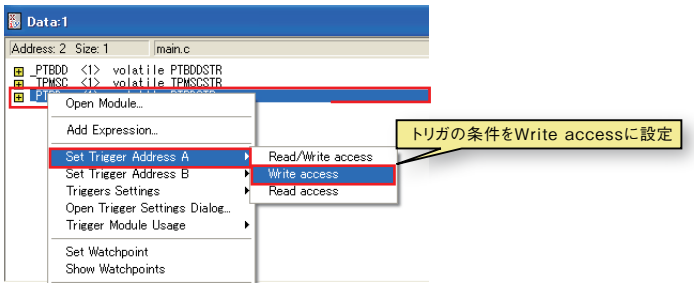


図4-12 CodeWarrior：メモリトリガの設定

4.6 メモリを参照する

メモリ画面のアドレスの上で右クリックし、ポップアップメニューで各設定を行います。

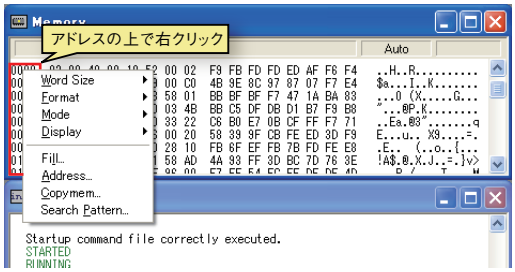


図4-13 CodeWarrior：メモリの参照

表4-1 メモリ参照メニュー

メニュー		設定内容	デフォルト
Word Size	Byte	バイト単位で表示	●
	Word	ワード（2 バイト）単位で表示	
	LWord	ロングワード（4 バイト）単位で表示	
Format	Hex	16 進数で表示	●
	Oct	8 進数で表示	
	Bin	2 進数で表示	
	Dec	10 進数で表示	
	UDec	符号付 10 進数で表示	
	BitReverse	ビット反転で表示（上記進数設定と組み合わせる）	
Mode	Automatic	プログラムが停止するたびにデータ表示が更新される	●
	Periodical	プログラムが実行中に一定時間ごとにデータ表示が更新される（デフォルトで 1 秒、100ms から 60 秒まで 100ms 単位で設定可）	
	Frozen	プログラムが停止してもデータ表示は更新されない	
Display	Address	Memory 画面左にアドレスを表示する	●
	ASCII	Memory 画面右にデータを ASCII に変換して表示する	●
Fill…		開始アドレスから終了アドレスにデータを設定する	-
Address…		表示するデータのアドレスを設定する	-
Copymem…		設定した領域のデータをコピーする	-
Search Pattern…		設定したパターンのデータを検索する	-

編集したいメモリをダブルクリックすると編集モードになります。

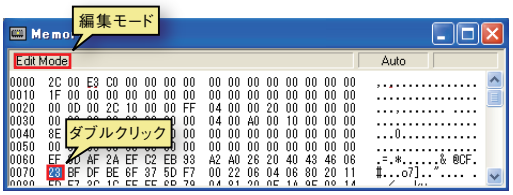


図4-14 CodeWarrior：メモリの編集

4.7 プロジェクトを閉じる

「Project」ウィンドウの「×」をクリックしてください。保存が必要な場合はメッセージが出ますので、「OK」をクリックしてください。

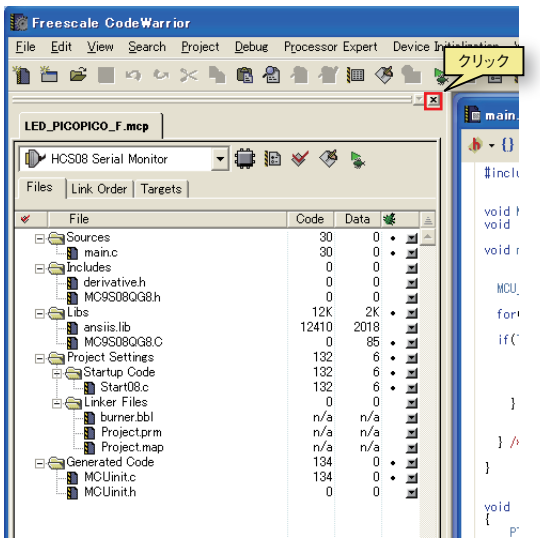


図4-15 CodeWarrior：プロジェクトを閉じる

4.8 プロジェクトを開く

「File」メニューの「Open…」をクリックするか、「Open…」ボタン をクリックしてください。プロジェクトフォルダ内の「プロジェクト名 .mcp」ファイルを選択し「開く」をクリックしてください。

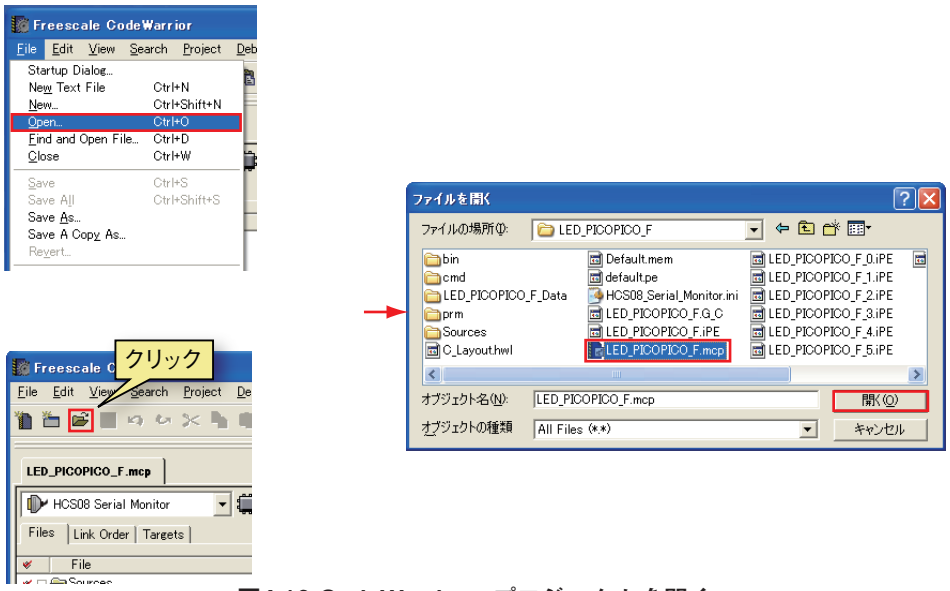


図4-16 CodeWarrior：プロジェクトを開く

CodeWarrior では複数のプロジェクトを開くことができます。各プロジェクトのコンパイル、デバッガ起動は各「Project」ウィンドウのツールバーより行ってください。

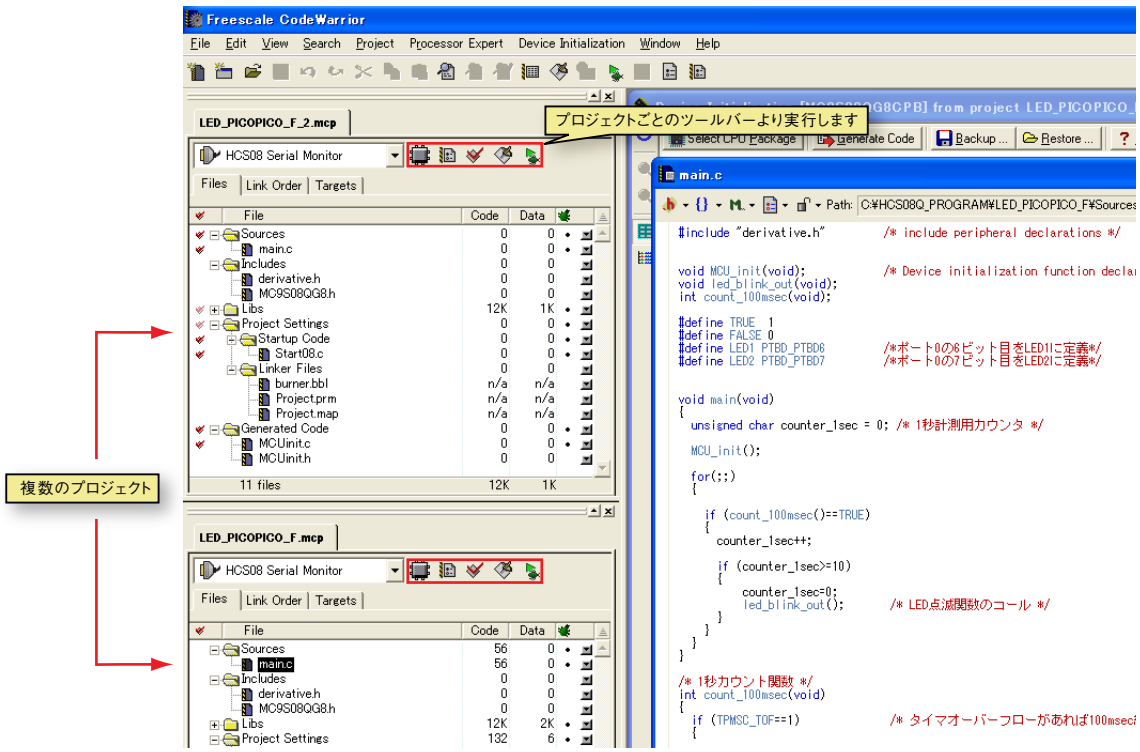


図4-17 CodeWarrior：複数のプロジェクトを開く

5. トラブルシューティング

5.1 デバッガと通信できない

■シリアル・モニタがデバッグモードで動作していない

SF9S08C マイコンのリセット解除時に評価カード CT-298 の SW1 が押されていないと、シリアル・モニタはデバッグモードで動作しません。通信できない場合は、USB コネクタを一度外してから再度 SW1 を押しながら USB コネクタを接続してください。

■COMポート番号が合っていない

デバッガで設定されている COM ポート番号と、ホスト PC で割り当てられている仮想 COM ポート番号が合っていない場合は、「5.2 仮想 COM ポート番号の確認」を参考に仮想 COM ポート番号を確認し、正しい COM ポート番号を設定してください。

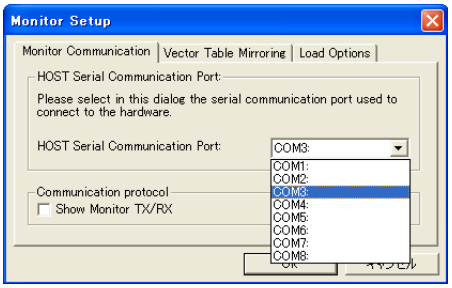


図5-1 COMポート番号の指定

■仮想COMポート番号が設定可能範囲以外の場合

デバッガでは COM ポート番号は 9 以上は設定できません。また 3 軸加速度センサーデモプログラムの評価アプリケーションでは COM ポート番号は 5 以上は設定できません。

「5.3 仮想 COM ポート番号の割り当て」を参考に仮想 COM ポートを範囲内の番号に割り当ててください。

5.2 仮想 COM ポート番号の確認

以下の方法で仮想 COM ポート番号を確認してください。

① Windows の「マイ コンピュータ」を右クリックし、「プロパティ」をクリックしてください。

② 「システムのプロパティ」ウィンドウの「ハードウェア」タブをクリックし、「デバイスマネージャ」をクリックします。

③ 一覧より「ポート（COM と LPT）」の「＋」マークをクリックすると、現在仮想 COM ドライバに割り当てられている COM ポートの番号が確認できます。

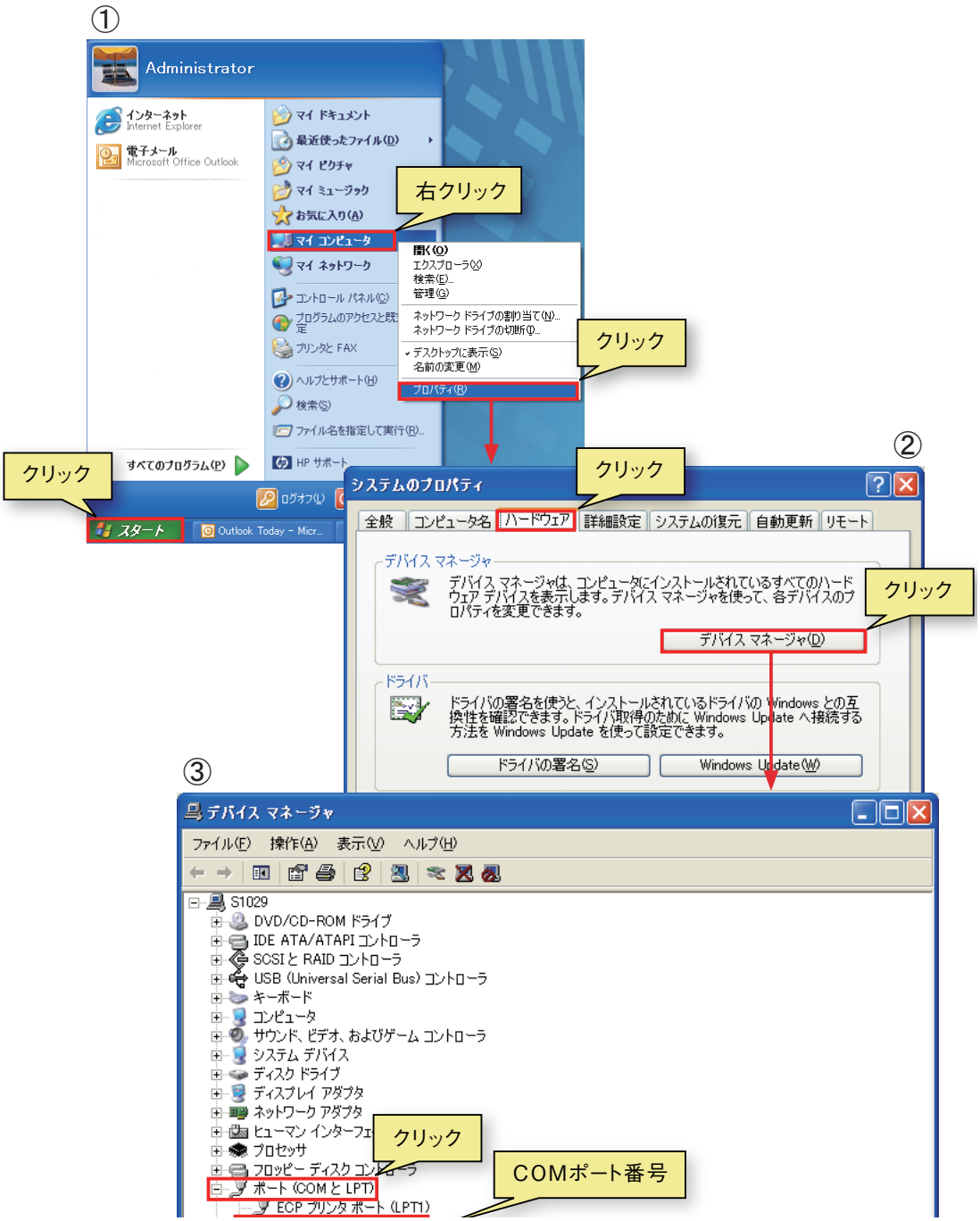


図5-2 仮想COMポートの割り当ての確認

5.3 仮想 COM ポート番号の割り当て

以下の方法で仮想 COM ポートのポート番号を再割り当てしてください。

① 「5.2 仮想 COM ポート番号の確認」を参考にデバイスマネージャを開き、現在割り当てられている COM ポートを開いてください。

② 仮想 COM ポートをダブルクリックし、プロパティを開いて「Port Settings」をクリックしてください。

③ 「Advanced…」をクリックして、「Advanced Settings」ウィンドウより、“(in used)”となっていない番号を選んでください。選択したら「OK」をクリックして「デバイスマネージャ」のウィンドウに戻ってください。

※ “(in used)”となっている番号でも実際は使用されていない場合がありますので、現在ホスト PC に他の仮想 COM ポートを使用するデバイスが接続されていないか確認の上選択してください。

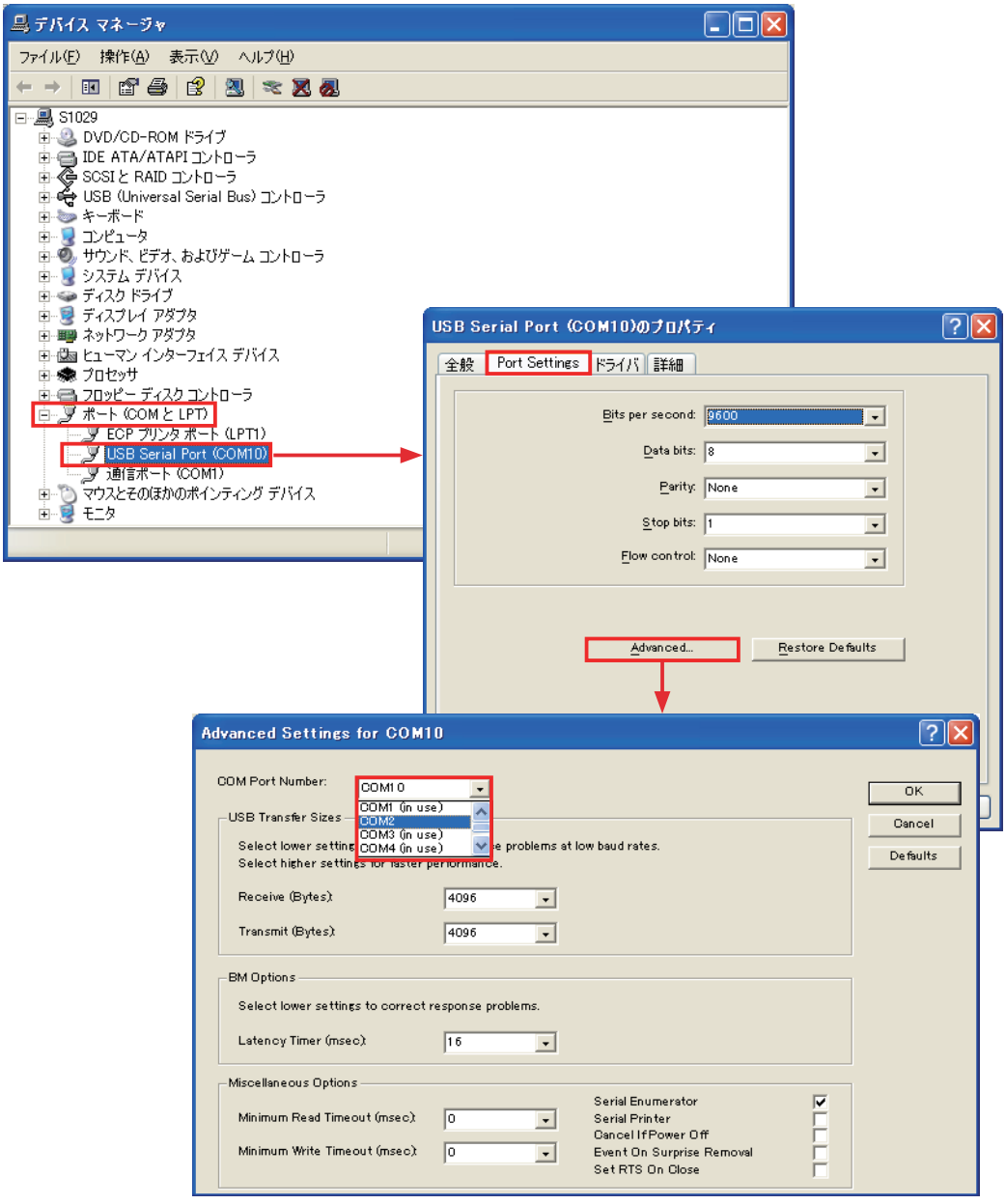


図5-3 仮想COMポートの再割り当て

改訂履歴

Rev.	発行日	ページ	改訂内容
1.00	2006/05/20	-	初版発行
1.01	2006/06/05	5,15,18	CodeWarrior Version5.0 から Version5.1 への変更についての注意事項を追加
1.10	2007/04/15	-	CodeWarrior Version5.0 から Version5.1 へ各記述、画像を変更 （hsc08.ini ファイルの更新を削除）
		13	Freescale 社のサイト構成変更に伴うダウンロード手順、画像を修正
		14	コラム「仮想 COM ポートドライバについて」を追加
		28	コラム「内部クロックについて」を追加

HC08 マイコンシリーズ
スタートアップガイド　シリアル・モニタ編

発行日　2007 年 04 月 15 日 Rev1.10
発　行　サンハヤト株式会社　〒 170-0005 東京都豊島区南大塚 3 丁目 40 番 1 号
©2006 Sunhayato Corp. All rights reserved.

SG043220